AÑO II - NUM. 53

MICROHOBBY

REVISTA INDEPENDIENTE PARA USUARIOS DE ORDENADORES SINCLAIR

SEMANAL

AÑO II - N.º 53

125 PTS.
Canarias 135 ptas.

NUEVO

"WEST BANK"

ATRACO EN EL LEJANO OESTE

CODIGO MAQUINA

INSTRUCCIONES ARITMETICAS Y LOGICAS

MICROFILE

COMO PASAR PROGRAMAS DE CINTA A MICRODRIVE

Página 24

UTILIDADES

UNA RUTINA PARA SIMPLIFICAR Y COMPRIMIR GRAFICOS

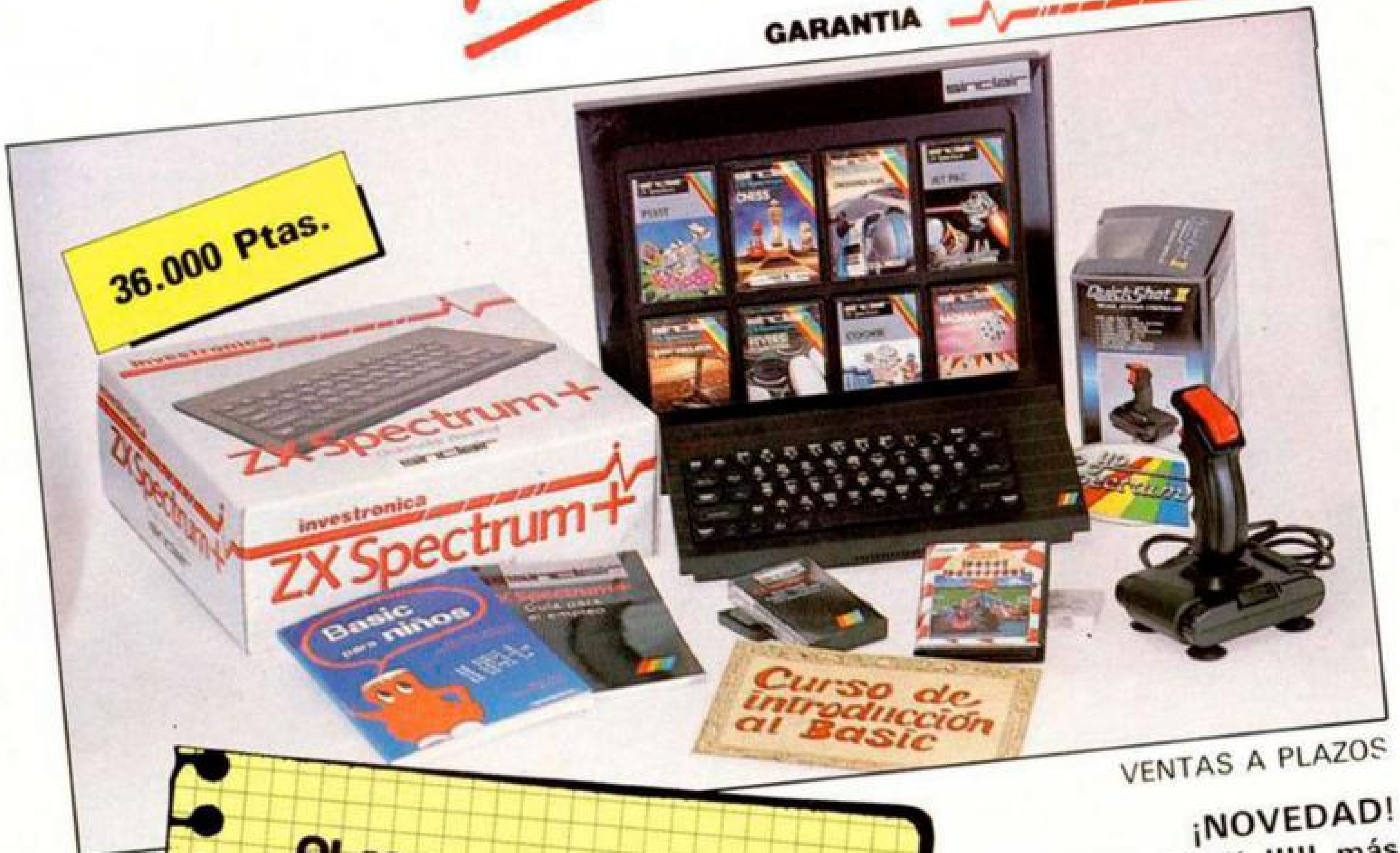


HOBBY PRESS, S.A.

COMPUTIQUE

Te da más

GARANTIA investronica



VENTAS A PLAZOS

Al comprar tu Spectrum
te regalamos

- Joystick Gunshot I
- Estuche con ocho juegos originales
  - — Backgammon
  - — Reversi
  - — Chess
  - — Chequered Flag
  - — Cookie
  - — Psst
  - — Jet Pac
  - — Flicht Simulation
- Un estupendo libro de Basic para niños.
- El mejor programa «Pole Position»
- Y además te obsequiamos con un curso de introducción al Basic

¡NOVEDAD!
Spectrum 128 K IIIII más memoria, dos teclados y en castellano.
Compatible con Spectrum 48 K/Plus
¡Más de 5.000 programas!



COMPUTIQUE

Servimos a tiendas
Abrimos sábados por la tarde

Embajadores, 90 Tfno. 2270980
28012 Madrid

Director Editorial
José I. Gómez-Centurión
Director Ejecutivo
Domingo Gómez
Asesor Editorial
Gabriel Nieto
Redactor Jefe
Africa Pérez Tolosa
Diseño
Rosa María Capitel
Redacción
Amalio Gómez,
Pedro Pérez
Jesús Alonso
Secretaria Redacción
Carmen Santamaría
Colaboradores
Primitivo de Francisco,
Rafael Prades,
Miguel Sepúlveda,
Sergio Martínez y J. M. Lazo
Fotografía
Javier Martínez, Carlos Candel
Portada
José María Ponce
Dibujos
J. R. Ballesteros, A. Perera,
F. L. Frontán, Pejo, J. M. López
Moreno J. Igual, J. A. Calvo, Lóriga,
J. Olivares
Edita
HOBBY PRESS, S. A.
Presidente
María Andrino
Consejero Delegado
José I. Gómez-Centurión
Jefe de Publicidad
Marisa Esteban
Publicidad Barcelona
José Galán Cortés
Tels.: 303 10 22 - 313 71 76
Secretaria de Dirección
Marisa Cogorro
Suscripciones
M.ª Rosa González
M.ª del Mar Calzada
Redacción, Administración y Publicidad
La Granja, s/n
Polígono Industrial de Alcobendas
Tel.: 654 32 11
Telex: 49480 HOPR
Dto. Circulación
Carlos Peropadre
Distribución
Coedis, S. A. Valencia, 245
Barcelona
Imprime
Rotedic, S. A. Ctra. de Irún,
km. 12,450 (MADRID)
Fotocomposición
Espacio y Punto, S. A.
Paseo de la Castellana, 268
Fotomecánica
Grof
Ezequiel Solana, 16
Depósito Legal
M-36.598-1984
Representante para Argentina, Chile, Uruguay y Paraguay, Cia. Americana de Ediciones, S.R.L. Sud América 1.532. Tel.: 21 24 64. 1209 BUENOS AIRES (Argentina).

Solicitado control OJD

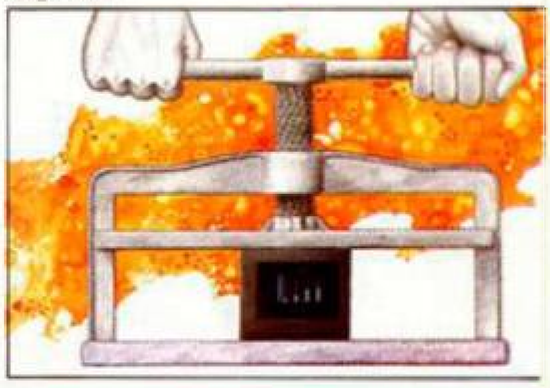
MICROHOBBY
ESTA SEMANA

AÑO II. N.º 53. 19 al 25 de noviembre de 1985
125 ptas. (Sobretasa Canarias 10 ptas.)

4 MICROPANORAMA.
7 TRUCOS.
8 PROGRAMAS MICROHOBBY. Montecarlo.
12 NUEVO Siéntete en el antiguo Oeste con la última aventura de Dinamic, «West Bank».
16 HARDWARE. El Microprocesador Z-80 (III)
19 CODIGO MAQUINA Instrucciones aritméticas y lógicas.
24 MICROFILE. Cómo pasar programas de cinta a Microdrive.
26 PROFESOR PARTICULAR.
29 EXPANSION.
30 UTILIDADES.
34 CONSULTORIO.
38 OCASION.



UTILIDADES. Una rutina para simplificar y comprimir gráficos. Pág. 30.



PREMIADOS HOBBY-SUERTE

JAVIER SAN JURGO PAZ. Pantaleón, 76. Betanzos (LA CORUÑA).
Suscripción a Microhobby Semanal por un año (4.ª Cat.)

MIGUEL MARIN. S. Pedro, 7. Guadalupe (MURCIA).
Una cinta de programas (5.ª Cat.)

JOSE LUIS BERMECHEA TEJERO. Ayola, 8, 2.º (VALENCIA).
Una cinta de programas (5.ª Cat.)

JOSE LUIS SALCIS BALLESTEROS. Capitán Haya, 56 (MADRID).
Un Joystick con su Interface (3.ª Cat.)

JUAN JOSE ALONSO GORDO. Seco, 3. Ciudad de Barcelona (MADRID).
Impresora Seikosha (2.ª Cat.)

AGUSTIN VALCARCEL POLANCO. Alta, s/n. Palacio de Justicia (SANTANDER).
Una cinta de programas (5.ª Cat.)

JOSE LUIS CID COHI. Pedro Cornell, 2, 7º Almazora (CASTELLON).
Cinta de programas (5.ª Cat.)

JOSE M.ª SANCHEZ ROMERA. S. Isidro Labrador, 56, 2.º (PALMA DE MALLORCA).
Cinta de programas (5.ª Cat.)

BENJAMIN MONTERRUBIO PUENTE. Ramón Patuel, 3 (MADRID).
Un Joystick con su Interface (3.ª Cat.)

FCO. JAVIER GARCIA GARCIA. Avda. César Augusto, 17-19 (ZARAGOZA).
Suscripción a Microhobby Semanal por un año (4.ª Cat.)

M.ª PAZ TALAVERA DIAZ. Soberanía Nacional, 7. Añover de Tajo (TOLEDO).
Cinta de programas (5.ª Cat.)

JOSE FUENTES JAUNSARAS. Rafaela Bonilla, 4 (MADRID).
Suscripción a Microhobby Semanal por un año (4.ª Cat.)

SALOMON DELGADO. Urb. Las Torres, 5, 1.º A (BURGOS).
Cinta de programas (5.ª Cat.)

JOSE MONNE ESQUERDA. Segocia, 10, 4.º (MERIDA).
Suscripción a Microhobby Semanal por un año (4.ª Cat.)

BLAS VAQUERO LOPEZ. Villalcampo, 1 (MADRID).
Una impresora Seikosha (2.ª Cat.)

EL DINAMISMO DE DINAMIC



Dinamic, la casa española de software con más repercusión nacional e internacional, no para de moverse y de crear nuevos programas.

Por el momento tienen ya casi ultimadas dos nuevas creacions: Sgrizam y Olé Toro, y otro más, Camelot Warriors se encuentra en un estado bastante avanzado de realización.

El primero en aparecer en el mercado, y además muy próximamente, será Sgrizam (al cual corresponden las fotos que os presentamos). El juego consiste básicamente en encontrar un maravilloso tesoro y rescatar a la princesa Doxapin, que se encuentran encerrados en el castillo de los malvados Kindos, todo ello envuelto en un ambiente entre futurista y medieval.

En cuanto a Camelot Warriors, aún tenemos muy pocas noticias, pero sabemos que consistirá en que un valeroso guerrero del siglo XIII deberá localizar una serie de objetos que provenientes del futuro, han llegado misteriosamente a su época: el Elixir de la vida, la Voz de otro mundo, el Fuego que no quema y el Espejo de la sabiduría.

Pero las actividades de Dinamic no acaban aquí. Además de poner en marcha un nuevo concurso: la Operación Puzzle, con el cual se pueden obtener programas de regalo con el simple hecho de unir dos piezas para formar su anagrama, Dinamic tiene aún otro proyecto de lo más ambicioso.

Consiste en la creación de una nueva casa de software que llevará el nombre de Future Star, para lo cual están intentando reunir a un buen número de programadores nobeles. Si crees que tienes aptitudes, ponte en contacto con ellos, quizás algún día te conviertas en un famoso programador...







Olé Toro, como su nombre indica clarísimamente, versará sobre el tema taurino, y acerca de él os podemos adelantar que los señores de Dinamic se han asesorado con los mayores expertos en temas relacionados con el mundo de los toros y por todo ello, les ha quedado un programa, nunca mejor dicho, redondo.

NOVEDADES SINCLAIR PARA NAVIDADES

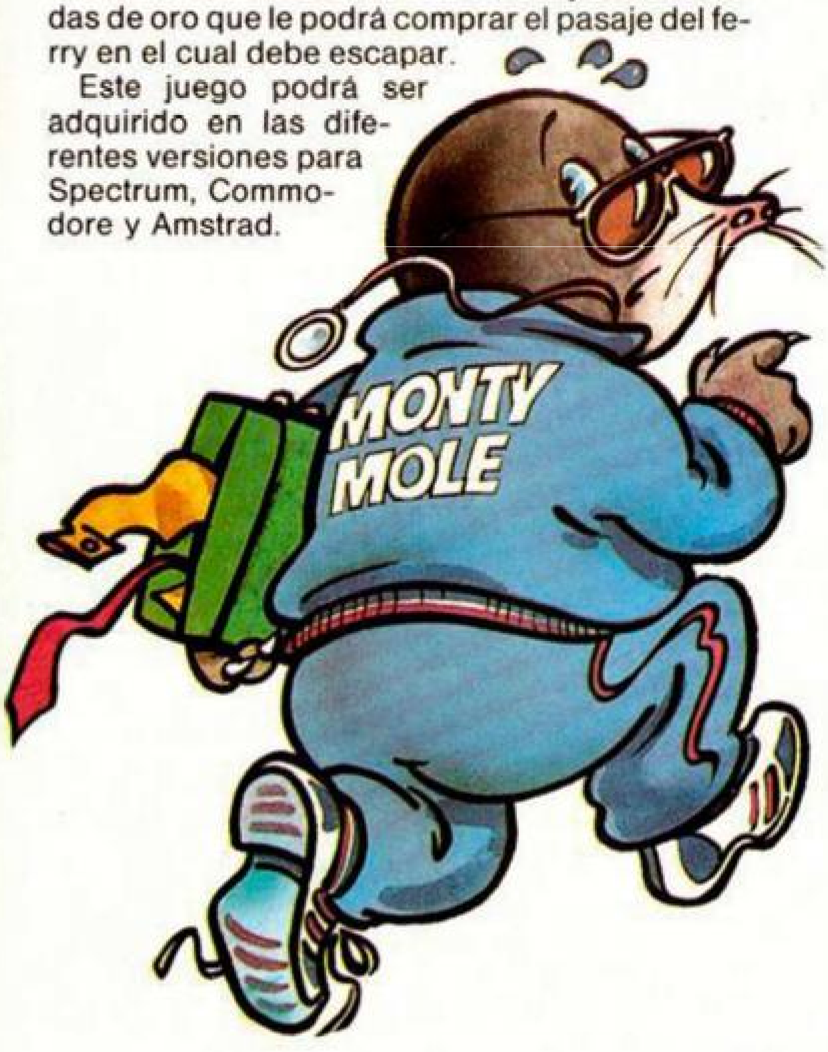
Con vistas a las próximas fechas navideñas, Sinclair tiene preparados tres nuevos productos para lanzar al mercado: el portátil Pandora, el Spectrum 128 y una nueva versión del QL, el QLII. Este último tendrá 256 ó 512 K y posiblemente llevará incorporado en la ROM todo el software de Psion que anteriormente se presentaba en un estuche con cuatro cartuchos de microdrive.

MONTY CONTINUA EN APUROS

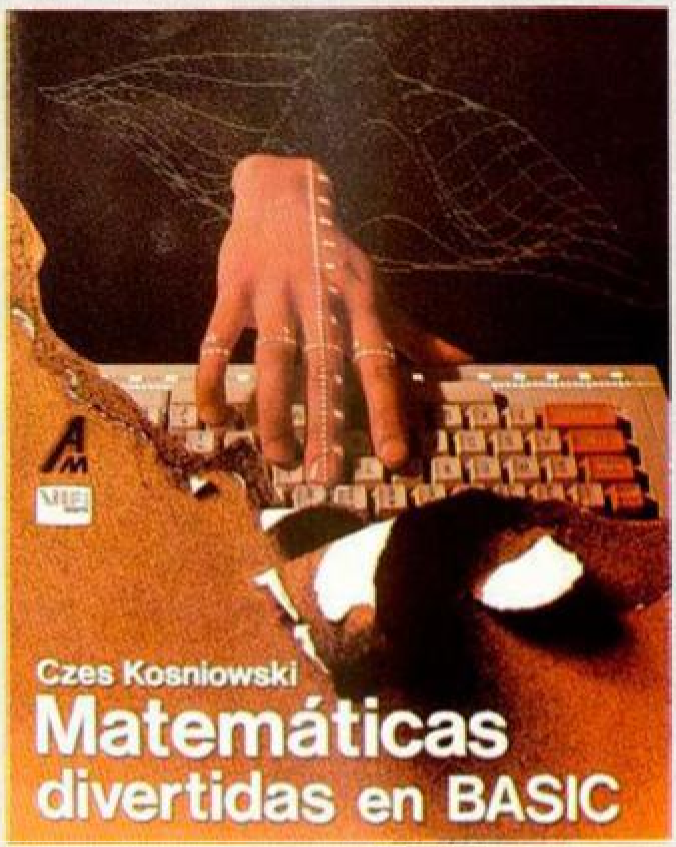
Gremlin Grafics, creadores del Abu Simbel Profanation, han lanzado una segunda parte de Monty is Inocent titulada Monty on the run.

En esta ocasión el simpático oso escapa de la prisión en la que se encuentra encerrado. Perseguido por la justicia, Monty se ve obligado a refugiarse en una guarida junto a otros criminales y su misión consiste en seleccionar de entre los muchos objetos que se le ofrecerán, aquéllos que le permitan conseguir su libertad definitiva, todo esto sin olvidar en su escondrijo las monedas de oro que le podrá comprar el pasaje del ferry en el cual debe escapar.

Este juego podrá ser adquirido en las diferentes versiones para Spectrum, Commodore y Amstrad.



LIBROS



MATEMATICAS DIVERTIDAS EN BASIC. Czes Kosniowski

Anaya Multimedia. 280 páginas

Si te interesan las matemáticas, aquí tienes un libro con el que podrás conocer más y mejor este fascinante mundo con la ayuda de tu ordenador.

El libro está compuesto por diez capítulos diferentes en los que se incluyen diversos temas relacionados con actividades humanas, pero enfocadas desde el punto de vista de las matemáticas. En la mayoría de las ocasiones estos temas son tratados de una manera simpática y muy asequible para cualquier lector, bien en forma de juegos o de pasatiempos, acompañados todos ellos de un buen número de dibujos.

Los programas utilizan para su realización el lenguaje Basic específico para el Apple II, pero al final del libro se incluyen unos pequeños apéndices donde se listan los programas adaptados especialmente para el Spectrum y para el Sharp MZ-731, además de algunas breves notas para otras posibles adaptaciones.

Los capítulos, que son totalmente independientes unos de otros, presentan temas tan importantes como el estudio de las series y sucesiones, funciones, matrices, grupos, ecuaciones diferenciales, etc, que comienzan con una ligera introducción matemática, para pasar posteriormente a la realización de algunos programas, por lo general bastante largos y con un nivel relativamente alto, que sirven como ilustración y complemento del tema central.

Debemos señalar que no está destinado a enseñar Basic, sino a aprender a utilizarlo como método de ayuda para resolver ciertas cuestiones matemáticas, por lo tanto este libro está orientado a aquellas personas que tengan un conocimiento medio de ambos temas.

TRUCOS

RAICES PARA TODOS LOS GUSTOS

José Manuel Alarcón ha efectuado un pequeño programa con el que se pueden hallar todo tipo de raíces, permitiéndote salir de la rutina de la raíz cuadrada.

```
   5 BORDER 2: INK 5: PAPER 3: C
LS
  10 REM * RAICES *
  20 PRINT "© PROUCTED BY J. M.
ALARCON": BEEP 1,8
  30 PRINT : BEEP .1,3
  40 PRINT : BEEP .1,2
  50 INPUT "INTRODUCE UN NUMERO:
 ";R: BEEP 1,7
  60 PRINT : BEEP .1,1
  70 PRINT : BEEP .1,0
  80 INPUT "INTRODUCE EL NUMERO,
 QUE SERA EL INDICE DE LA RAIZ:
 ";S: BEEP 1,6
  90 PRINT R↑1/S: BEEP 1,5
 100 GO TO 30
```



A. PERERA

JUGAR CON LOS BORDERs

Para conseguir diferentes efectos con los BORDERs, no tienes más que teclear estas rutinas que nos manda José Domingo Cerdán.

Si, además, se le pone «CLS» detrás de cada border, se obtienen unos colores diferentes.

```
  10 REM "Borders de varios colo
res"
  20 BORDER 1: BORDER 2: BORDER
3: BORDER 4: BORDER 5: BORDER 6
  30 GO TO 20
```

```
  10 BORDER 1: BORDER 2
  20 GO TO 10
```

```
   5 GO TO 20
  10 BORDER 1: BORDER 4
  15 RETURN
  20 GO SUB 10
  30 GO TO 5
```

```
  10 REM "Papers de varios color
es"
  20 PAPER 1: CLS : PAPER 2: CLS
 : PAPER 3: CLS : PAPER 4: CLS :
 PAPER 5: CLS : PAPER 6: CLS
  30 GO TO 20
```

ESPACIO EN BLANCO

Con este truco de Santiago Castaño puedes conseguir eliminar espacios en blanco sobrantes de las sentencias DIM, cosa que, como verás te será de gran utilidad y ahorro.

```
  10 BORDER 2: PAPER 6: CLS
  12 PRINT TAB 2;"ELIMINACION DE
 LOS ESPACIOS";TAB 1;"SOBRANTES
DE LAS SENTENCIAS DIM"
  13 PRINT ''" 1. Linea 2130 Buc
le descendente del valor de la s
entencia DIM"
  14 PRINT ''" 2. Linea 2140  Se
 cuentan los        espacios sobra
ntes de la          variable a$(i)
 "
  15 PRINT ''" 2. Linea 2150 La
variable a$(i) se convierte en d
$"
  20 INPUT "Relacion ";N;" Espac
ios de cada una ";M
  22 DIM a$(n,m): DIM a(n)
 106 CLS : PRINT TAB 12;"Pregunt
as"''
 112 FOR i=1 TO N
 120 INPUT " ";(i);" ";a$(i): GO
 SUB 2100
 124 PRINT TAB 2,i; PAPER 4;" ";
d$
 136 NEXT i
 194 PAUSE 0: STOP
2110 REM ***********************
           ESPACIOS EN BLANCO
2120 FOR j=i TO i
2130 FOR f=M TO 1 STEP -1
2140 IF a$(j)(f TO f)=" " THEN L
ET a(j)=a(j)+1: NEXT f
2150 IF a$(j)(f TO f)<>" " THEN
LET d$=a$(j)( TO LEN a$(j)-a(j))
: NEXT j: RETURN
```

MONTECARLO

Matti HEMMI

Spectrum 48 K

¿Te gusta el juego? ¿Has soñado alguna vez con disfrutar del lujo y el ambiente de un casino tan internacional como es el de Montecarlo? Pues con un «mucho» de imaginación y este programa, lo conseguirás.



Se trata de apostar una cantidad determinada a los números y letras de la ruleta que te muestra el ordenador. Si, una vez parada, la aguja se situara ante tu apuesta, ganarás el triple, si no ganará la Banca.

Puedes apostar hasta ocho veces con tansólo mover el asterisco situado en la parte superior derecha de la pantalla, usando las teclas 5 y 8, apretando, finalmente, la P cuando hayas elegido el número deseado.

¡Buena suerte!

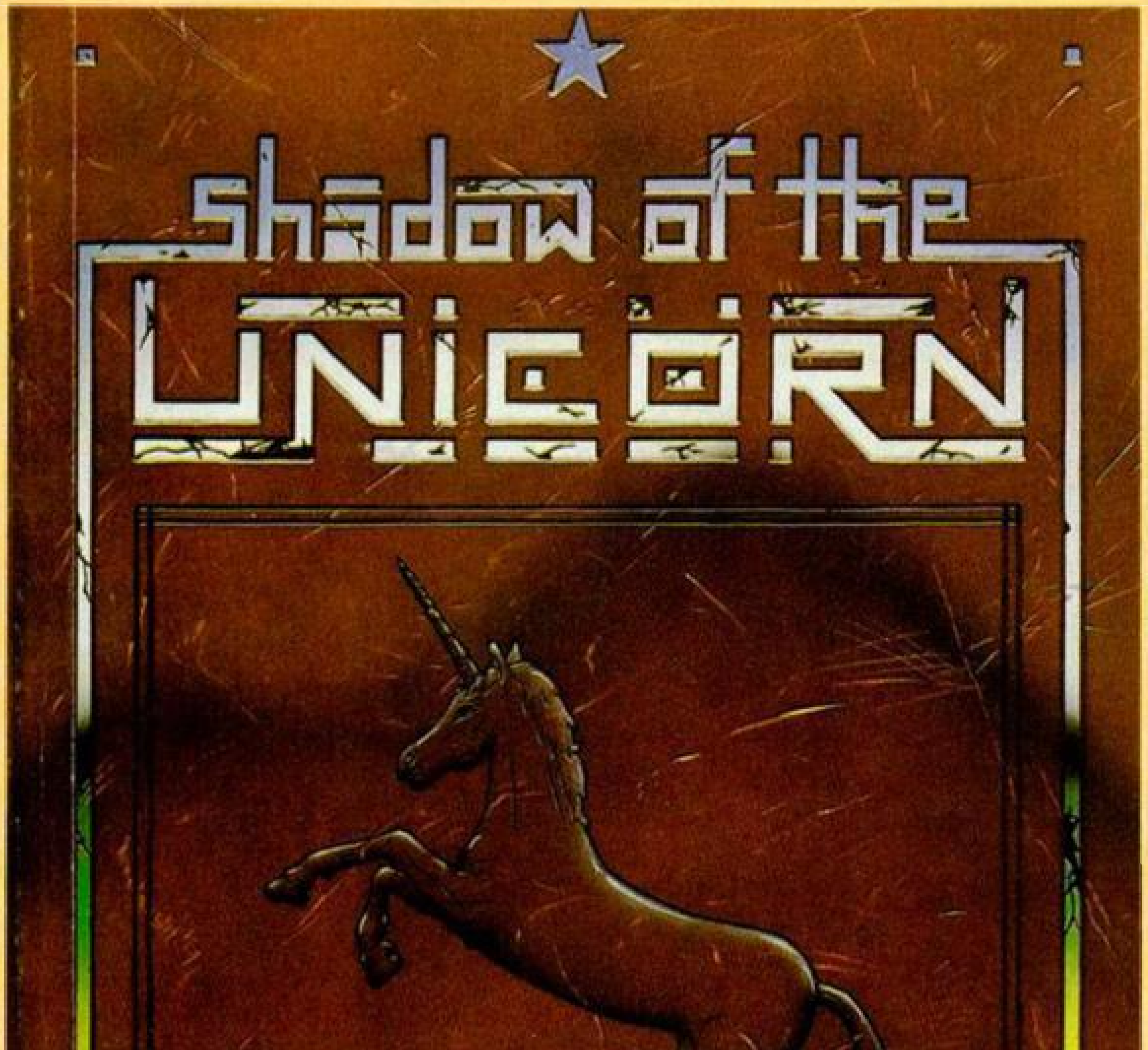
```
   6 RANDOMIZE : POKE 23658,8
   7 INK 7: PAPER 0: BORDER 0: C
LS
   8 LET tot=200: LET total=tot:
 LET gi=0: FOR n=1 TO 12: PRINT
AT 11,n-2;" ": PAUSE n: BEEP .03
,2*n: PRINT AT 11,n;" RULETA ZX"
;AT n-2,13;"          ";AT n-1,13
;"RULETA ZX": NEXT n: PAUSE 40
   9 LET a1=0: LET a2=0: LET a3=
0: LET a4=0: LET a5=0: LET a6=0:
 LET a7=0: LET a8=0: LET p1=0: L
ET p2=0: LET p3=0: LET p4=0: LET
 p5=0: LET p6=0: LET p7=0: LET p
8=0: LET h=0: LET ko=0: LET m=0:
 LET n=0: LET pa=.5: LET i=0: LE
T t=0: LET x=20: LET y=2: LET co
n=0: IF gi=1 THEN GO TO 17
  10 LET k1=119: LET k=SGN PI: L
ET x$="
        PRUEBA TU SUERTE EN LA RU
LETA ZX.PULSA ""P"" PARA EMPEZAR
 A JUGAR,""I"" PARA INSTRUCCIONE
S;
"
  11 PRINT #NOT PI; PAPER 2;AT N
OT PI,NOT PI;x$(k TO k+31): IF k
=k1 THEN LET k=SGN PI
  12 IF INKEY$="P" THEN GO TO 17
  13 IF INKEY$="I" THEN CLS : GO
 TO 16
  14 BEEP .1,2: LET k=k+SGN PI:
GO TO 11
  16 GO SUB 8000
  17 LET gi=1: INK 7: PAPER 4: B
ORDER 4: CLS : LET tot=total: RE
STORE : GO SUB 19
  18 GO TO 200
  19 REM ***TABLERO APUESTAS***
  20 PLOT 156,163: DRAW 96,0: DR
AW 0,-128: DRAW -96,0: DRAW 0,12
8
  25 FOR l=1 TO 7: PLOT 156,163-
(l*16): DRAW 96,0: NEXT l
  30 FOR k=1 TO 3: PLOT 156+(k*2
4),163: DRAW 0,-128: NEXT k
  31 FOR n=2 TO 8 STEP 2: FOR m=
21 TO 30 STEP 3: READ k$: PRINT
AT n,m;k$: NEXT m: NEXT n
  34 RESTORE : FOR n=10 TO 16 ST
EP 2: FOR m=21 TO 30 STEP 3: REA
D k$: PRINT INK 4; PAPER 7;AT n,
m;k$: NEXT m: NEXT n
  35 PRINT AT 0,21;"1  2  3  4":
 FOR m=2 TO 16 STEP 2: READ s$:
PRINT AT m,18;s$: NEXT m
  36 GO SUB 6060
  37 REM ***DIBUJO RULETA***
  40 CIRCLE 72,102,72: CIRCLE 72
,102,55
  50 RESTORE 9040: FOR i=1 TO 8:
 READ p: READ q: READ r: READ s:
 PLOT p,q: DRAW r,s: NEXT i
  60 FOR i=1 TO 32: READ n: READ
 m: READ v$: PRINT AT n,m;v$: NE
XT i
  70 RETURN
 199 REM ***MOVIMIENTO AGUJA***
 200 FOR t=1 TO 64+INT (RND*32)
STEP 2
 210 LET a=t/16*PI
 220 LET sx=72*SIN a: LET sy=72*
COS a
 230 BEEP .03,003: PLOT 72,102:
DRAW OVER 1;sx,sy
 235 IF h=6 THEN LET pa=pa+2: LE
T h=0
 240 PAUSE pa
 250 PLOT 72,102: DRAW OVER 1;sx
,sy
 255 LET h=h+1
 260 NEXT t
 270 PLOT 72,102: DRAW OVER 1;sx
,sy: BEEP .3,24: BEEP .45,4
2900 REM ***COMPROBANDO ACIERTO***
2993 IF t>=67 THEN LET t=t-64: G
O TO 2997
2995 IF t>=34 THEN LET t=t-32
2997 LET k=t
3000 LET n$=STR$ (k/2-.5)
3011 IF n$="10" THEN LET n$="A"
3012 IF n$="11" THEN LET n$="B"
3013 IF n$="12" THEN LET n$="C"
3014 IF n$="13" THEN LET n$="D"
3015 IF n$="14" THEN LET n$="E"
3016 IF n$="15" THEN LET n$="F"
3017 IF n$="16" THEN LET n$="G"
3025 IF n$>"9" THEN LET c$="LA":
 GO TO 3050
3030 LET c$="EL"
3050 PRINT AT 19,3;"LA AGUJA SE
DETUVO EN ";c$;" ";n$;AT 20,6;"P
ARA SABER SU COLOR";AT 21,8;"PUL
SA UNA TECLA": PAUSE 0: PRINT AT
 19,3;"

                             "
3060 LET az=INT (RND*2): IF az=1
 THEN LET t$="BLANCO": GO TO 306
7
3063 LET az=0: LET t$="VERDE"
3067 LET j=k/2-.5
3068 IF az=1 THEN LET po=j: GO T
O 3074
3070 LET po=j+16
3074 IF (po=a1) OR (po=a2) OR (p
o=a3) OR (po=a4) OR (po=a5) OR (
po=a6) OR (po=a7) OR (po=a8) THE
N PRINT FLASH 1;AT 20,6;" HAS TE
NIDO SUERTE";AT 21,2;"UNA DE TUS
 APUESTAS ACERTO": PRINT #0;"
     PULSA UNA TECLA": PAUSE 0: G
O TO 3100
3080 PRINT AT 19,3;"EL PREMIO RE
CAYO EN ";c$;" ";n$;AT 20,6;"DE
COLOR ";t$: PRINT #0;"       PULS
A UNA TECLA": PAUSE 0: CLS : GO
TO 3190
3090 REM ***REPARTO PREMIO***
3100 LET pre=(p1 AND po=a1)+(p2
AND po=a2)+(p3 AND po=a3)+(p4 AN
D po=a4)+(p5 AND po=a5)+(p6 AND
po=a6)+(p7 AND po=a7)+(p8 AND po
=a8): GO TO 3110
3110 LET total=tot+3*pre
3120 CLS : PRINT AT 2,0;"TENIAS
";tot;" FICHAS TRAS LA ULTIMA";A
T 3,0;" APUESTA.AHORA TIENES ";t
otal;" FICHAS": GO TO 3200
3190 IF tot=0 THEN GO TO 3290
3192 PRINT AT 2,3;"TRAS LAS ULTI
MAS APUESTAS";'';"    TIENES ";tot
;" FICHAS": LET total=tot
3200 PRINT AT 6,3;"SI NO QUIERES
 VOLVER A PROBAR";'';"   SUERTE P
ULSA LA ""N"" SI NO";'';"    PULSA
 CUALQUIER TECLA": PAUSE 0: CLS
3210 IF INKEY$<>"N" THEN GO TO 9
3220 CLS : GO TO 3220
3290 CLS : PRINT AT 15,0;" SE AC
ABO,YA NO TIENES DINERO.";'';"SI
TE QUIERES ARRIESGAR A JUGAR";'"O
TRA PARTIDA,PULSA CUALQUIER";'"TE
CLA,DE LO CONTRARIO PULSA";'"LA "
"N""."; PAUSE 0: IF INKEY$<>"N"
THEN LET total=200: CLS : GO TO
9
3295 CLS : GO TO 3295
6050 REM ***APUESTAS***
6060 PRINT AT 0,0;"RECUERDA:";AT
 2,0;"""5"" IZQUIERDA";AT 3,0;""
"8"" DERECHA";AT 4,0;"""P"" APOS
TAR";AT 5,0;"""C"" CAMBIAR";AT 6
,0;"""R"" EMPEZAR";AT 9,4;"FICHA
S";AT 10,3;"APOSTADAS": PLOT 18,
83: DRAW 80,0: DRAW 0,26: DRAW -
80,0: DRAW 0,-26
6100 PRINT AT 19,23;"TIENES ";AT
 20,22-LEN STR$ tot;"  ";AT 20,2
4-LEN STR$ tot;tot;AT 20,24;" FI
CHAS"
6102 DIM i(8): DIM f$(8)
6103 LET y=2: LET x=20
6104 PRINT AT y,x;"*"
6110 LET o=4: LET u=7: LET e=x:
LET d=y: IF con=8 THEN GO TO 680
0
6120 IF INKEY$="5" THEN GO TO 61
60
6130 IF INKEY$="8" THEN GO TO 61
90
6140 IF INKEY$="P" THEN LET con=
con+1: GO TO 6215
6147 IF INKEY$="R" THEN GO TO 68
00
6150 GO TO 6110
6154 IF INKEY$="C" THEN LET tot=
tot+apu: GO TO 6310
6159 GO TO 6110
6160 IF x<=20 AND y<>2 THEN LET
x=29: LET y=y-2: GO TO 6205
6170 IF y=2 AND x=20 THEN GO TO
6110
6180 LET x=x-3: GO TO 6205
6190 IF x>=29 AND y<>16 THEN LET
 x=20: LET y=y+2: GO TO 6205
6200 IF y=16 AND x=29 THEN GO TO
 6110
6201 LET x=x+3
6205 PRINT AT d,e;" "
6210 PRINT OVER 1;AT y,x;"*": GO
 TO 6110
6215 IF ATTR (y,x+1)=23 OR ATTR
(y,x+1)=20 THEN LET con=con-1: G
O TO 6110
6220 INPUT "CUANTO APUESTAS ? ";
apu: IF apu<0 OR apu>tot THEN GO
 TO 6220
6223 LET tot=tot-apu
6225 PRINT AT 19,23;"TIENES ";AT
 20,22-LEN STR$ tot;"  ";AT 20,2
4-LEN STR$ tot;tot;AT 20,24;" FI
CHAS": IF ko=1 THEN LET ko=0: GO
 TO 6320
6226 IF y>=10 THEN GO TO 6230
6228 LET i(con)=(1 AND y=2 AND x
=20)+(2 AND y=2 AND x=23)+(3 AND
 y=2 AND x=26)+(4 AND y=2 AND x=
29)+(5 AND y=4 AND x=20)+(6 AND
y=4 AND x=23)+(7 AND y=4 AND x=2
6)+(8 AND y=4 AND x=29)+(9 AND y
=6 AND x=20)+(10 AND y=6 AND x=2
3)+(11 AND y=6 AND x=26)+(12 AND
 y=6 AND x=29)+(13 AND y=8 AND x
=20)+(14 AND y=8 AND x=23)+(15 A
ND y=8 AND x=26)+(16 AND y=8 AND
 x=29)
6229 GO TO 6231
6230 LET i(con)=(17 AND y=10 AND
 x=20)+(18 AND y=10 AND x=23)+(1
9 AND y=10 AND x=26)+(20 AND y=1
0 AND x=29)+(21 AND y=12 AND x=2
0)+(22 AND y=12 AND x=23)+(23 AN
D y=12 AND x=26)+(24 AND y=12 AN
D x=29)+(25 AND y=14 AND x=20)+(
26 AND y=14 AND x=23)+(27 AND y=
14 AND x=26)+(28 AND y=14 AND x=
29)+(29 AND y=16 AND x=20)+(30 A
ND y=16 AND x=23)+(31 AND y=16 A
ND x=26)+(32 AND y=16 AND x=29)
6231 IF i(con)>16 THEN LET o=7:
LET u=4: GO TO 6235
6233 LET o=4: LET u=7
6235 IF con=1 THEN LET a1=i(con)
: LET p1=apu: PRINT AT 12,0;"  "
;AT 12,3-LEN STR$ p1;p1;" FICHAS
 A ";: PAPER o; INK u;SCREEN$ (y,
x+1): LET f$(1)=SCREEN$ (y,x+1)
6236 IF con=2 THEN LET a2=i(con)
: LET p2=apu: PRINT AT 13,0;"  "
;AT 13,3-LEN STR$ p2;p2;" FICHAS
 A ";: PAPER o; INK u;SCREEN$ (y,
x+1): LET f$(2)=SCREEN$ (y,x+1)
6237 IF con=3 THEN LET a3=i(con)
: LET p3=apu: PRINT AT 14,0;"  "
;AT 14,3-LEN STR$ p3;p3;" FICHAS
 A ";: PAPER o; INK u;SCREEN$ (y,
x+1): LET f$(3)=SCREEN$ (y,x+1)
6238 IF con=4 THEN LET a4=i(con)
: LET p4=apu: PRINT AT 15,0;"  "
;AT 15,3-LEN STR$ p4;p4;" FICHAS
 A ";: PAPER o; INK u;SCREEN$ (y,
```



```
x+1): LET f$(4)=SCREEN$ (y,x+1)
6239 IF con=5 THEN LET a5=i(con)
: LET p5=apu: PRINT AT 16,0;"  "
;AT 16,3-LEN STR$ p5;p5;" FICHAS
 A "; PAPER o; INK v;SCREEN$ (y,
x+1): LET f$(5)=SCREEN$ (y,x+1)
6240 IF con=6 THEN LET a6=i(con)
: LET p6=apu: PRINT AT 17,0;"  "
;AT 17,3-LEN STR$ p6;p6;" FICHAS
 A "; PAPER o; INK v;SCREEN$ (y,
x+1): LET f$(6)=SCREEN$ (y,x+1)
6241 IF con=7 THEN LET a7=i(con)
: LET p7=apu: PRINT AT 18,0;"  "
;AT 18,3-LEN STR$ p7;p7;" FICHAS
 A "; PAPER o; INK v;SCREEN$ (y,
x+1): LET f$(7)=SCREEN$ (y,x+1)
6242 IF con=8 THEN LET a8=i(con)
: LET p8=apu: PRINT AT 19,0;"  "
;AT 19,3-LEN STR$ p8;p8;" FICHAS
 A "; PAPER o; INK v;SCREEN$ (y,
x+1): LET f$(8)=SCREEN$ (y,x+1)
6245 PRINT OVER 1;AT y,x+1;"  "
6250 PAUSE 0: IF tot=0 AND INKEY
$<>"C" THEN GO TO 6800
6255 GO TO 6154
6310 INPUT "CUANTO QUIERES APOST
AR AHORA ?  ";apu: IF apu<0 OR a
pu>tot THEN GO TO 6310
6315 LET ko=1: GO TO 6223
6320 GO TO 6235
6790 REM ***FICHAS APOSTADAS***
6800 CLS : PLOT 163,130: DRAW 80
,0: DRAW 0,26: DRAW -80,0: DRAW
0,-26: PRINT AT 3,22;"FICHAS";AT
 4,21;"APOSTADAS": FOR k=1 TO co
n: READ z
6803 IF i(k)>16 THEN LET o=7: LE
T v=4: GO TO 6810
6804 LET o=4: LET v=7
6810 PRINT AT 6+k,21-LEN STR$ z;
z;" FICHAS A "; PAPER o; INK v;f
$(k)
6820 NEXT k
6900 RETURN
8000 REM ***INSTRUCCIONES***
8010 LET a$="ESTE JUEGO CONSISTE
 EN APOSTAR*FICHAS A LOS NUMEROS
 Y LETRAS*QUE APARECERAN EN LA R
ULETA ZX.*SI LA AGUJA SE PARASE
EN ALGUN*NUMERO O LETRA DE LOS Q
UE HAS*APOSTADO,GANARAS EL TRIPL
E DE*LO APOSTADO.PERO SI NO OCUR
RE*ASI,GANARA LA BANCA ZX.*SE PU
EDEN HACER COMO MAXIMO UN*TOTAL
DE OCHO APUESTAS.PODRAS*APOSTAR
TANTAS FICHAS COMO*TENGAS.CUANDO
 NO DESEES APOSTAR*MAS,PULSA LA
""R""."
8011 LET w$="PARA APOSTAR DEBERA
S MOVER EL*ASTERISCO SITUADO EN
LA PARTE*SUPERIOR DERECHA DEL RE
CUADRO*QUE APARECERA AHORA.*PARA
 ELLO USA LAS TECLAS ""5""*Y ""8
"".CUANDO TENGAS COLOCADO*EL AST
ERISCO JUNTO A LA CASILLA*QUE QU
IERAS APOSTAR PULSA LA*""P"" Y A
 CONTINUACION INDICA*LA CANTIDAD
 QUE QUIERAS APOSTAR*PERO SI DES
EAS CAMBIAR ALGUNA*APUESTA PULSA
 LA ""C"".*SOLO PODRAS CAMBIAR L
A APUESTA*INMEDIATAMENTE DESPUES
 DE HACER*CADA UNA."
8014 FOR g=1 TO 375
8015 IF a$(g)="*" THEN LET m=m+1
: LET n=0: GO TO 8025
8017 LET n=n+1
8020 BEEP .02,03: PRINT AT m,n;a
$(g)
8025 NEXT g
8027 PRINT #0;"          PULSA UNA
 TECLA"
8030 PAUSE 0: CLS
8040 FOR r=1 TO 414
8050 IF w$(r)="*" THEN LET t=t+1
: LET s=0: GO TO 8080
8060 LET s=s+1
8070 BEEP .02,03: PRINT AT t,s;w
$(r)
8080 NEXT r
8090 PRINT #0;"          PULSA UNA
 TECLA"
8100 PAUSE 0: CLS : RETURN
9000 DATA "1","2","3","4","5","6
","7","8","9","A","B","C","D","E
","F","G"
9010 DATA "A","B","C","D","E","F
","G","H"
9020 DATA p1,p2,p3,p4,p5,p6,p7,p
8
9040 DATA 72,174,0,-144,101,168,
-57,-131,122,154,-100,-101,137,1
31,-131,-57,144,102,-143,0,137,7
4,-131,57,122,52,-100,101,101,37
,-57,131
9050 DATA 1,10,"■",3,10,"1",2,13
,"■",4,12,"2",4,15,"■",5,13,"3",
7,16,"■",8,14,"4",10,16,"■",10,1
4,"5",13,15,"■",12,13,"6",15,13,
"■",14,16,"7",17,10,"■",14,10,"8
",17,7,"■",14,7,"9",15,4,"■",14,
5,"A",13,2,"■",12,4,"B",10,1,"■"
,10,3,"C",7,1,"■",8,3,"D",4,2,"■
",5,4,"E",2,4,"■",4,5,"F",1,7,"■
",3,7,"G"
```

SI BUSCAS LO MEJOR



SHADOW OF THE UNICORN

UNA NUEVA ERA PARA TU SPECTRUM. UN PROGRAMA QUE INCORPORA UN INTERFACE AMPLIANDO LA MEMORIA A 64 K. CON SALIDA PARA JOYSTICK Y QUE TIENE ¡6.800 PANTALLAS! PUDIENDO CONTROLAR HASTA 10 PERSONAJES DIFERENTES.

¡UNA AUTENTICA REVOLUCION!



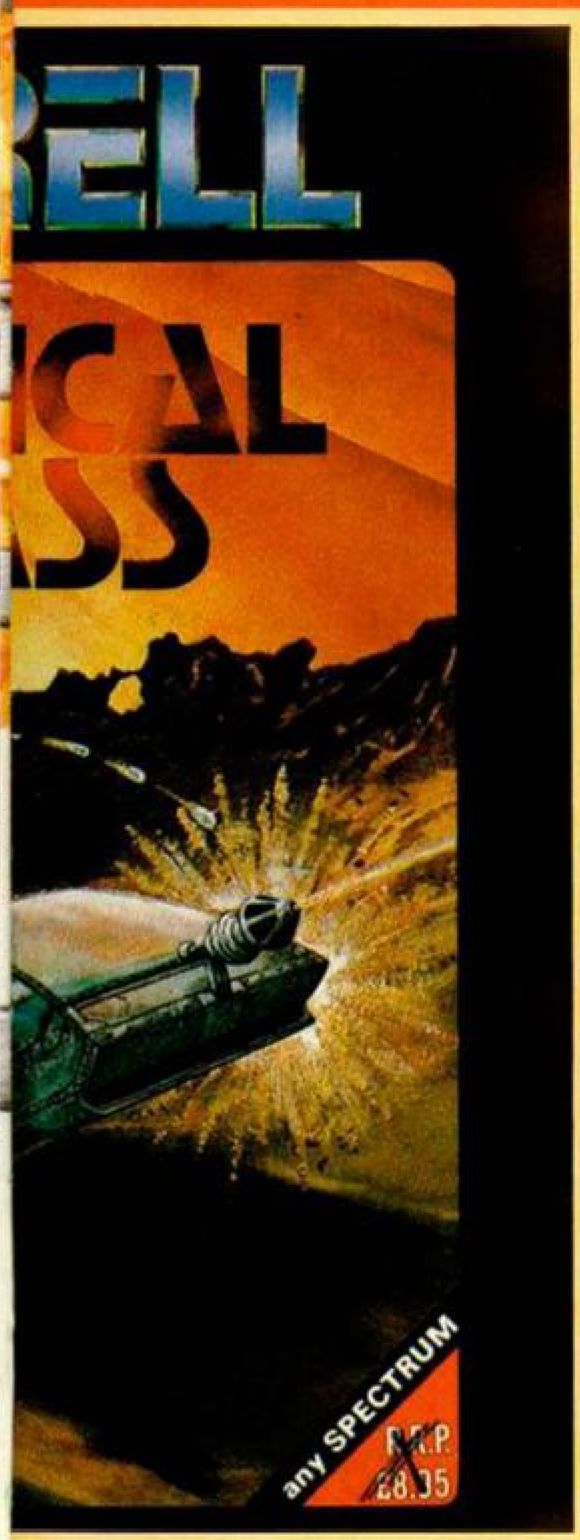
CRITICA

EL NUEVO Y ESPERADO JUEGO D
LANZO "SCUBA DIVE", "HARRI E
ENTRE OTROS.

UN JUEGO DE ACCION A TOPE
TODA TU HABILIDAD.

DISTRIBUCION EXCLUSIVA PA
SANTA ENGRACIA, 17 - TEL.

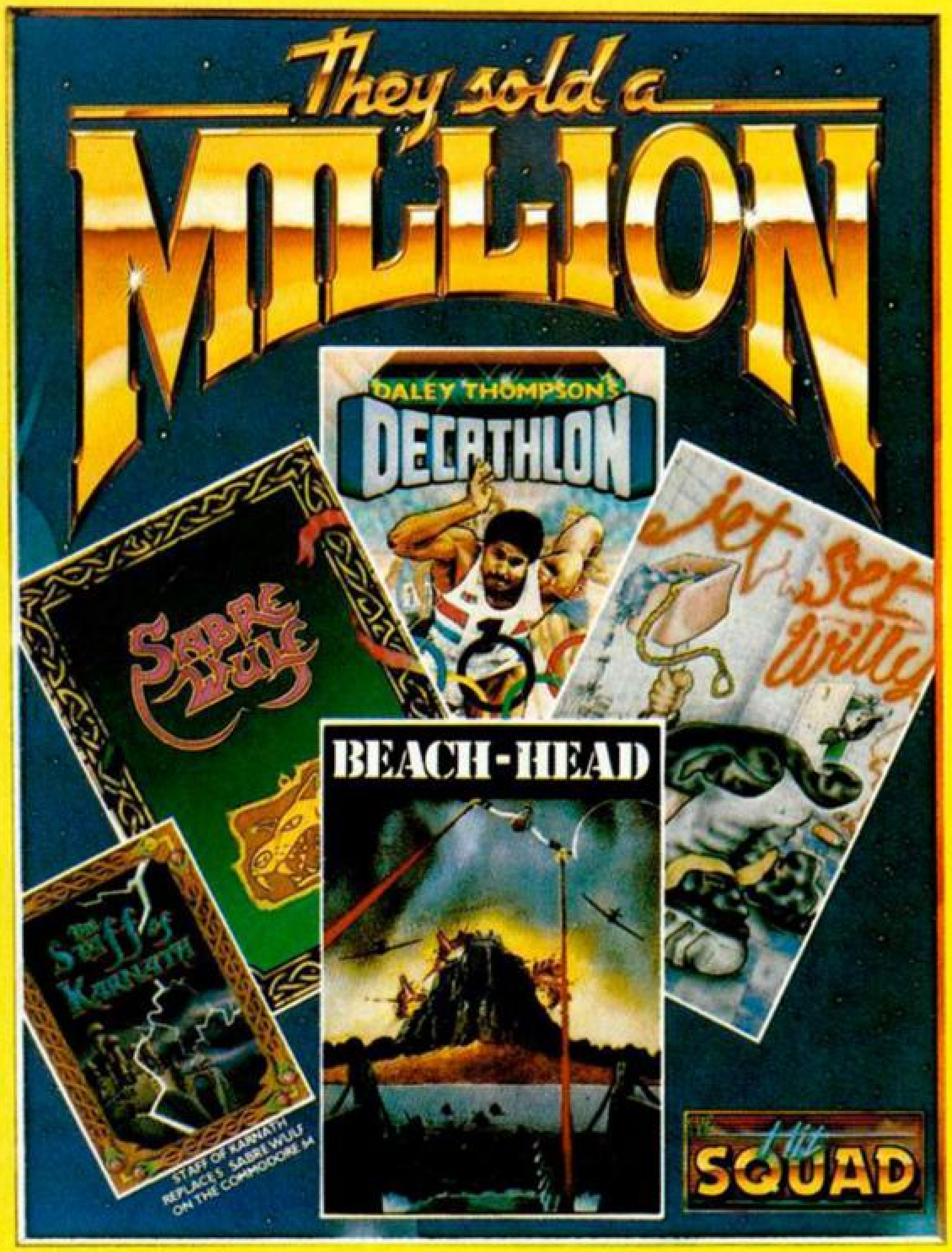
Software LO TIENE



L MASS

DE DURELL, LA COMPAÑIA QUE ER ATACK" Y "COMBAT LYNX"

EN EL QUE SERA NECESARIA



THEY SOLD A MILLION

LOS CUATRO MEJORES JUEGOS DE TODOS LOS TIEMPOS AL PRECIO DE 1.

SPECTRUM Y AMSTRAD: DECATHLON, JET SET WILLY, BEACH HEAD, SABREWULF

COMMODORE: STAFF ON KARNATH, BEACH HEAD, DECATHLON Y JET SET WILLY.

RA ESPAÑA ERBE SOFTWARE,

447 34 10 - 28010 MADRID

West Bank • Dinamic • Arcade

UNA HISTORIA DE DUELOS Y PISTOLEROS

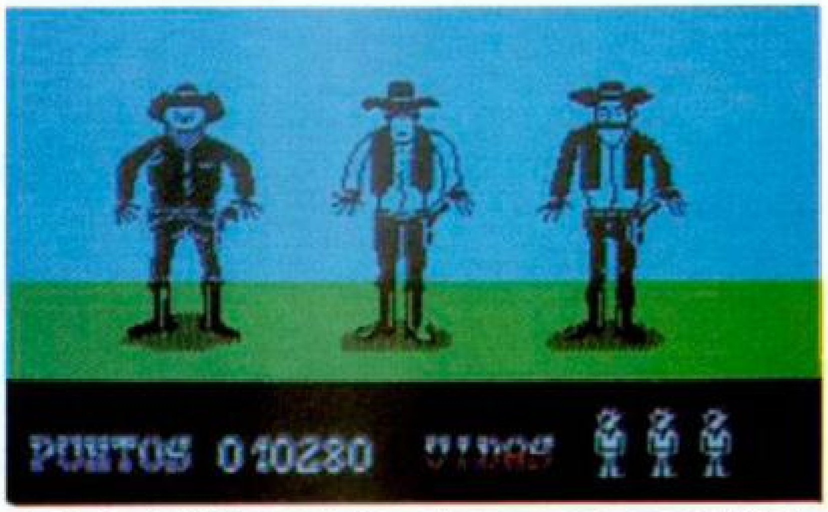
Dinamic presenta su última creación: West Bank, un divertido y emocionante juego en el que podrás poner a prueba tus reflejos y con el que tendrás la oportunidad de comprobar qué tal te hubiera ido como pistolero en el salvaje Oeste.

El West Bank es un banco muy especial. Recibe diariamente casi a tantos pistoleros dispuestos a atracarlo como pacíficos clientes que desean ingresar sus ahorros.

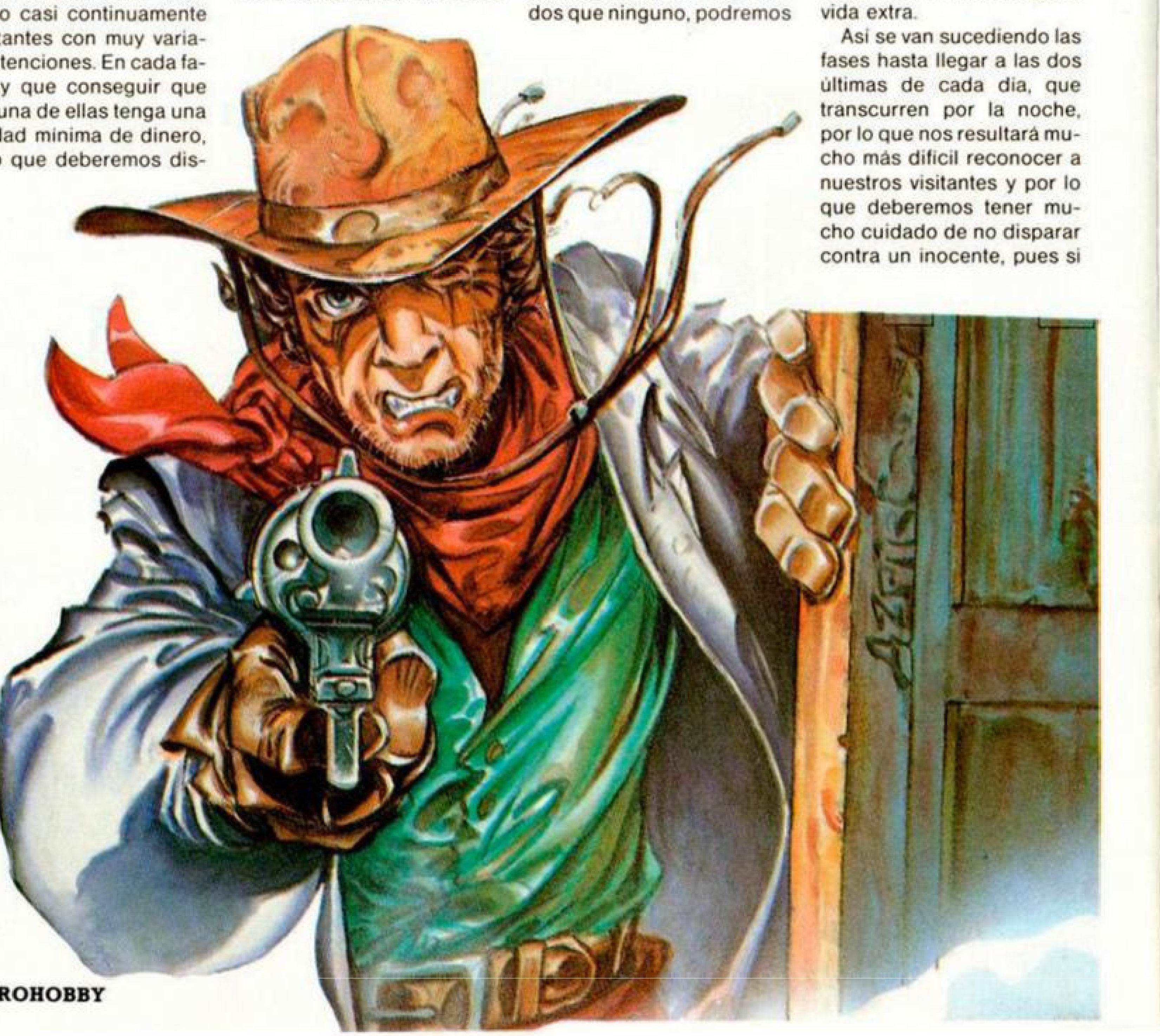
El juego consiste en mantener a raya a todos estos forajidos para permitir que las arcas del banco se vayan llenando poco a poco de dólares.

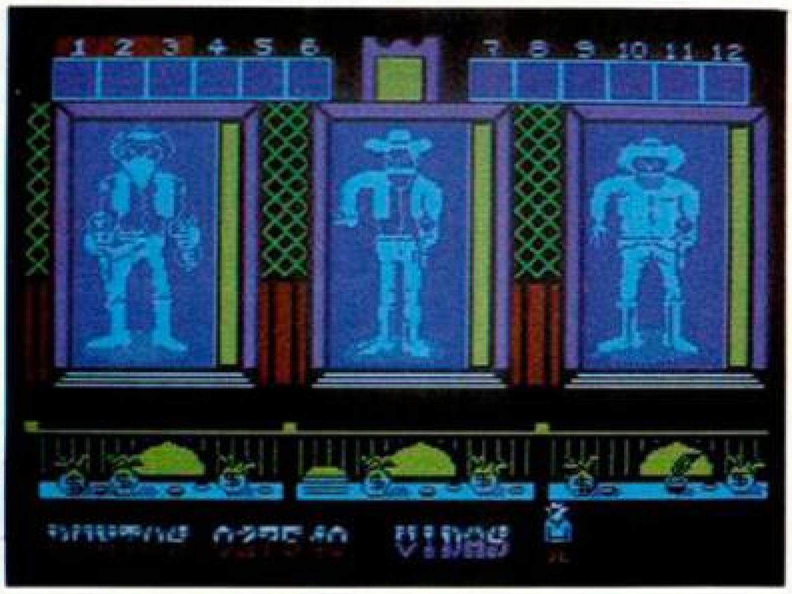
El banco está formado por doce habitaciones diferentes, las cuales están recibiendo casi continuamente a visitantes con muy variadas intenciones. En cada fase hay que conseguir que cada una de ellas tenga una cantidad mínima de dinero, por lo que deberemos disparar contra aquéllos que vengan a robar y dejar pasar a quien venga a ingresar sus dólares, pero hay que tener sumo cuidado de no disparar contra los pacíficos ciudadanos y de ser más rápidos en hacerlo que los atracadores.

Una vez conseguido que todas estas habitaciones tengan su cantidad correspondiente de dinero, deberemos enfrentarnos en un duelo a muerte con tres de los más temibles bandidos al Oeste del Mississipi. Si conseguimos ser más rápidos que ninguno, podremos obtener una buena cantidad de puntos e incluso, alguna vida extra.

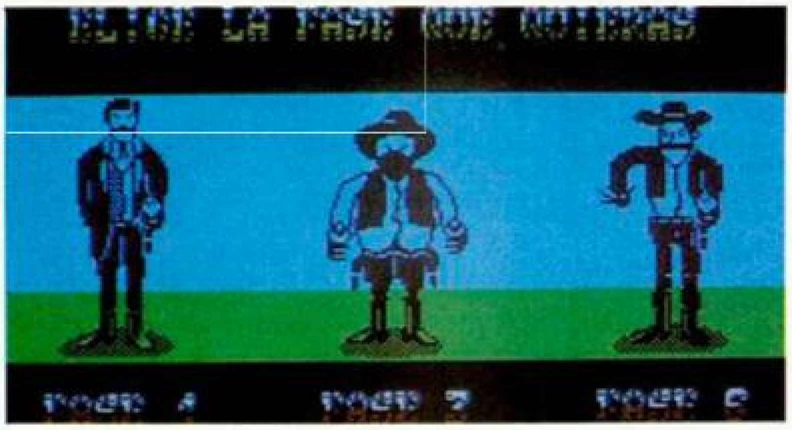


Así se van sucediendo las fases hasta llegar a las dos últimas de cada día, que transcurren por la noche, por lo que nos resultará mucho más difícil reconocer a nuestros visitantes y por lo que deberemos tener mucho cuidado de no disparar contra un inocente, pues si







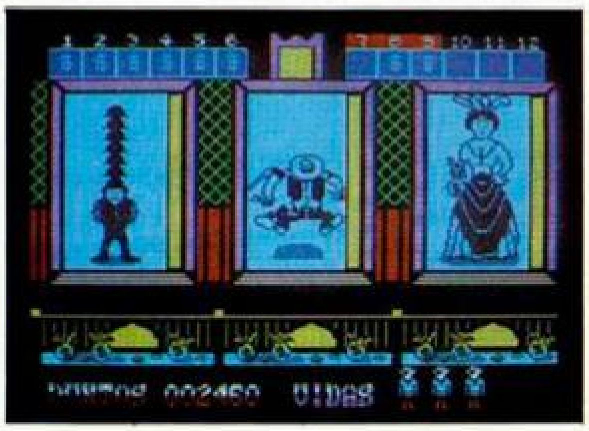


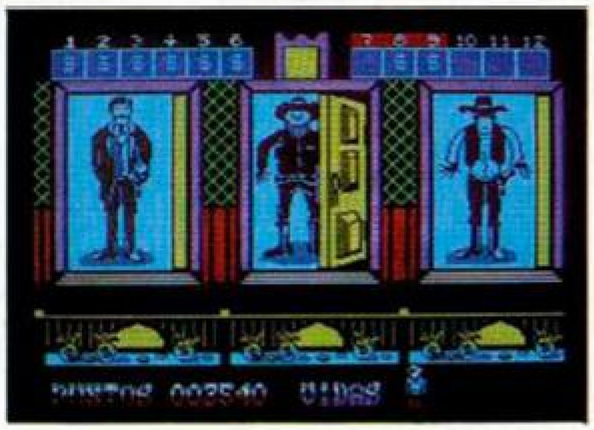
así lo hiciéramos, perderíamos una de nuestras valiosas vidas.

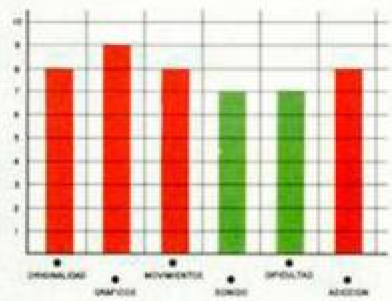
El juego resulta, en definitiva, muy entretenido y bastante emocionante, pues a medida que vamos avanzando en las fases y en los días, la cosa se va complicando: las puertas se van abriendo con mayor frecuencia y somos visitados por un número mayor de atracadores que, además, van siendo más rápidos en disparar, por lo que llega un momento en el que la tensión es máxima y no se permite ni el menor despiste, o de lo contrario... ¡Bang!, muerto.

Por otro lado, los gráficos son excelentes y los personajes representan fielmente a los forajidos de la época, por lo que jugar con este West Bank resulta una verdadera delicia.

Tan sólo tenemos que poner un pero a este juego, y es que quizá se eche de menos un poco de variedad en el desarrollo, pues debemos confesar que cuando llevas un buen rato dándole a las pistolas, resulta un pelín monótono; pero también es justo señalar que esto se suple con una creciente emoción y sin darte cuenta te vas «picando» cada vez más y te es difícil levantarte del sillón hasta no haber conseguido igualar las magníficas puntuaciones que consiguieron, muchos años atrás, pistoleros tan famosos como Lucky Lucke o Billy «El niño».







CONCURSO WEST BANK

Por si fueran pocos los alicientes de este West Bank, nosotros os ofrecemos aún uno más.

Como ya sabéis, el juego está dividido en varios días que a su vez se componen de nueve fases diferentes. Pues bien, si consigues llegar hasta la última fase del segundo día, en la pantalla aparece un pequeño mensaje junto con un dibujo.

¿Sabéis cuál es la frase? Pues si es así, enviadnos rápidamente una carta con la solución correcta indicando en el sobre «Concurso West Bank».

A las 25 primeras cartas recibidas les enviaremos gratuitamente un original de la próxima creación de Dinamic: «Sgrizam», a cuya realización aún se la están dando los últimos retoques. Sed los primeros en poseer este estupendo juego y..., ¡ya sabéis!, enviadnos lo más pronto posible vuestras respuestas.

Nombre y apellidos __________
Edad __________
Dirección __________
Población __________ Provincia __________
«La frase que aparece es» __________

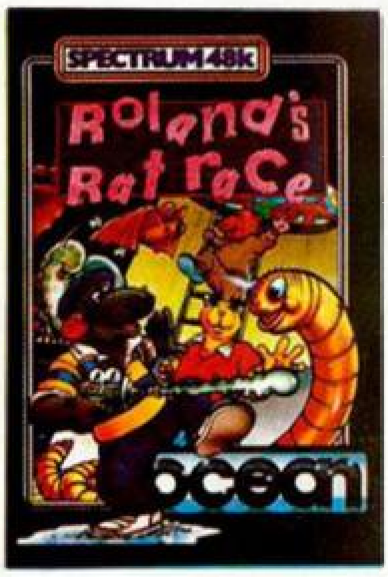
Recorta o fotocopia este cupón y envíalo a HOBBY PRESS, S.A. CONCURSO WEST BANK. c/La Granja, s/n. Polígono Industrial de Alcobendas. Alcobendas - MADRID.

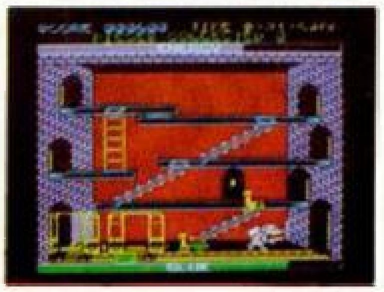
Una rata muy televisiva

ROLAND'S RAT RACE

Ocean

Arcade



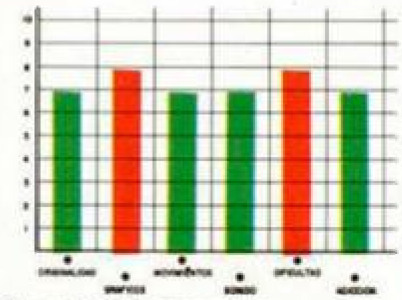


Ocean vuelve al mercado con un nuevo y divertido arcade. En esta ocasión el escenario son las cloacas y los túneles del metro de Londres y el protagonista es Roland, una simpática rata estrella de un programa matinal de la TV.



Con nuestra colaboración, Roland deberá ir recogiendo uno a uno los trozos con los que ha de formar una puerta para salir por ella y poder llegar a tiempo a los estudios de Televisión. Estos fragmentos se encuentran repartidos entre una maraña de pasadizos y túneles en los que resulta bastante difícil orientarse y por los cuales circulan, además, unas botas colocadas por nuestros enemigos –los envidiosos roedores que quieren quitarle el trabajo a Roland–, y que nos van restando la energía necesaria para poder conseguir nuestra misión. Para defendernos de ellas Roland lleva en su espalda un lanzador de pegamento que hace que éstas se queden adheridas al suelo y que así no obstaculicen nuestro camino. Cuando se nos acaben las reservas, tanto de pegamento como de energía, podremos ir consiguiendo más de los depósitos y alimentos variados que se encuentran repartidos por los intrincados pasadizos.

Una vez recompuesta toda la puerta, deberemos pasar por un pasadizo plagado de unos horribles murciélagos que nos dificultarán aún mucho más las cosas y que harán que resulte casi imposible que Roland llege a tiempo para su show.

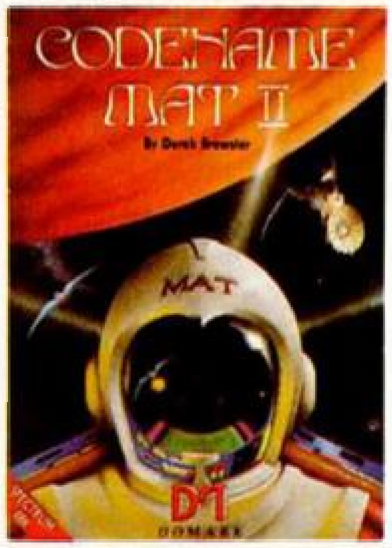


Salva a Vesta de la invasión alienígena

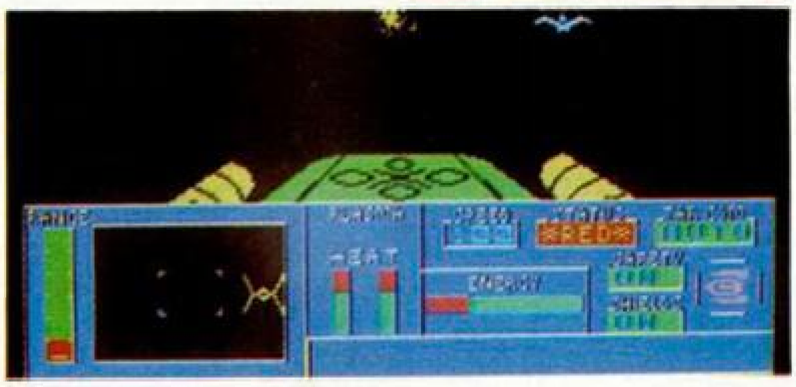
CONDENAME MAT II

Domark

Simulador



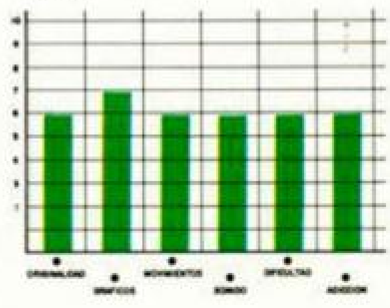
El planeta Vesta posee el producto más importante de todo el Universo: el Karilium. Durante muchos años la extracción y distribución de este producto se ha desarrollado con total normalidad, pero ahora los malvados Myrons han vuelto para destruir Vesta y sus minas de Karilium. MAT ha sido el piloto elegido para acabar con las naves de los Myrons y reestablecer la paz y el orden de nuevo en el espacio.



Las minas obtienen su energía de las estrellas Cales, la cual es enviada y transformada a través de una serie de satélites que son atacados por los Maryons. Tu misión es la de mantener alejadas las naves enemigas con el fin de que la producción de esta fundamental sustancia se vea lo menos obstaculizada posible, pues si todos los satélites son destruidos Vesta no podrá recibir la energía, las minas no podrán seguir produciendo, y el juego terminará.

Al final de cada ataque enemigo tendrás la oportunidad de reparar los satélites dañados y mover otros para recomponer las vías de transporte destruidas; pero para esta tarea deberás ser rápido y preciso, pues el más mínimo error podría suponer la victoria de los Maryons.

Recuerda que tú eres la única persona que puede defender Vesta, sin embargo, tu vida no correrá ningún peligro, pues si tu nave es destruida en alguno de los numerosos combates que tendrás que mantener, un sofisticado mecanismo de seguridad te teletransportará a algún lejano planeta del universo donde te encontrarás totalmente a salvo.





Recorte o copie este cupón y envíelo a Hobby Press, S.A., Apartado de Correos 54062 Madrid.

Deseo recibir en mi domicilio el ejemplar Especial n.º 1 de MICROHOBBY al precio de 300 ptas.

Nombre .... Apellidos ....

Edad .... Domicilio ....

Localidad .... Teléfono ....

Provincia .... Código Postal ....

El importe de este pedido lo abono de la siguiente forma:

- ☐ Mediante talón bancario adjunto a nombre de Hobby Press, S.A.
- ☐ Por giro postal n.º ....
- ☐ Contra reembolso del envío (supone 75 ptas. de gastos de envío)

Firma Fecha

EL MICROPROCESADOR Z-80 (III)

Primitivo de FRANCISCO

Tratamos en esta tercera parte sobre el registro de estado del Z-80 (también llamado registro de flags) y el significado de sus bits. Incluimos también una colección de los principales diagramas de tiempos del funcionamiento del Z-80. Estos gráficos han de ser de gran utilidad para los estudiosos de los montajes hardware así como para la comprensión del funcionamiento del Spectrum.

El registro de estado es una serie de ocho biestables, cada uno de ellos almacena temporalmente un bit. El significado de cada uno de estos bits está directamente relacionado con el funcionamiento interno del microprocesador. Su conocimiento es determinante para la comprensión de algunas de las instrucciones del Z-80 y por tanto para su programación.

Cuando se empieza a utilizar el código máquina pronto se ve la potencia funcional que tiene este registro el cual es laborioso acceder y modificar por software; ya que está concebido para que sea el propio microprocesador el que lo altere y, aunque puede ser leído por software, su cometido principal reside en que su estado es consultado por el Z-80 al ejecutar las instrucciones aritméticas, lógicas, etc.

De los ocho bits hay dos (el 3 y el 5) que no tienen significado para el programador, pero sí para el microprocesador, pues se apoya en ellos para la ejecución de ciertas operaciones internas. En principio los hemos excluido de nuestro estudio porque es el propio fabricante del Z-80 quien no suministra información oficial en absoluto sobre ellos. Aunque a base de laboriosos resultados empíricos hemos obtenido ciertos resultados que facilitaremos cuando los tengamos concluídos. Los seis bits restantes son utilizados por el microprocesador así:

- BIT Ø (C): CARRY (ACARREO). Este bit es puesto a uno por el Z-80 cuando el resultado de operar un dato rebosa en el acumulador el máximo valor posible con ocho bits 255 ó FF en hexadecimal, en caso contrario permanece a cero. Este bit puede ser afectado por las instrucciones aritméticas, lógicas, por las de rotación y desplazamiento. Si se operan datos de 16 bits con los registros dobles, entonces el bit C se pone a uno cuando el resultado de una operación excede a 65535 ó FFFF hexadecimal. Este bit también actúa de borrow o bit de acarreo en instrucciones de substracción.

– BIT 1 (N): ADD/SUBSTRACT (SUMA/RESTA EN BCD). Este bit es actuado en operaciones en BCD. BCD significa: Decimal Codificado en Binario en que obviamente están prohibidos los números hexadecimales A hasta F. Para trabajar en BCD se utilizan las mismas instrucciones que operan en hexadecimal, pero el resultado final requiere un ajuste decimal mediante la instrucción DAA, la cual utiliza la información de los bits 1 y 4 del registro de estado. En la operación de resta en BCD este bit se pone a 1 y a Ø en las de suma lógica, de rotación y desplazamiento.

– BIT 2 (P/V): PARITY/OVERFLOW (PARIDAD Y DESBORDAMIENTO). Este bit es utilizado como indicador de bit de paridad o como bit de desbordamiento según las instrucciones.

Las instrucciones aritméticas le usan como indicador de desbordamiento. Se produce desbordamiento cuando el resultado de una operación aritmética excede al rango entre –128 (8ØH) y +127 (7FH). En operaciones de ocho bits. Similarmente existe desbordamiento en operaciones de 16 bits cuando el resultado excede a rango –32768 (8ØØØ H) y +32767 (7FFFH).

Este bit del registro de estado es utilizado como bit de paridad en las instrucciones lógicas, de rotación y desplazamiento. Estas lo ponen a uno si en el resultado la suma de bits que están a 1 es par y a Ø si es impar.

– BIT 4 (H): HALF CARRY (ACARREO MEDIO INTERDIGITOS). Este bit tiene sólo sentido en operaciones en BCD en donde el acarreo que se produce cada cuatro bits es determinativo para su operatividad en sumas y restas. Recordemos que la instrucción DAA produce un ajuste decimal del resultado en hexadecimal en una operación aritmética. En BCD únicamente se utilizan las combinaciones binarias desde ØØØØ hasta 1ØØ1 (esto es desde Ø a 9) por tanto las otras seis combinaciones desde 1Ø1Ø hasta 1111 (desde A hasta F el hexadecimal) producen acarreo entre los dos dígitos BCD que entran en cada byte.

– BIT 6 (Z): ZERO (RESULTADO CERO). Este bit es alterado por las instrucciones aritméticas y lógicas y también por la de comprobación de bit.

En general el bit Z se pone a 1 si el resultado de una operación es Ø, y se pone a 1 cuando el resultado es distinto de cero. En este bit también se apoyan las instrucciones que implican decisión como son las de salto, que ejecutan o no la bifurcación según el resultado entregado en este bit producido por la instrucción inmediatamente anterior.

– BIT 7 (S): SIGN (SIGNO DEL RESULTADO). Este bit es alterado por las instrucciones aritméticas, lógi-



Fig. 1. Impulsos de reloj que precisa el Z-80 para los diferentes momentos clave en el funcionamiento de este microprocesador.

cas, de rotación y desplazamiento. Su significado consiste en indicar el signo del resultado de alguna operación de las indicadas. En el bit S del registro de estado queda copiado el bit de mayor peso de una operación. Así serán positivos los números comprendidos entre Ø y 7FH porque su bit número 7 es un cero, y negativos desde 8ØH hasta FFH porque su bit número 7 es un uno.

En las operaciones cuyo resultado implica una palabra ó 16 bit, el bit S se pone a uno en resultados comprendidos entre 8ØØØ H y FFFH, es decir, cuando el bit de mayor peso (el número 15) es un uno y son positivos entre ØØØØH y 7FFFH porque el bit 15 es un cero.

El registro de flags va unido al acumulador cuando ambos son utilizados como registro de 16 bits y se encuentra duplicado en ambos bancos de registros. Esto es útil a la hora de guardar temporalmente su contenido para luego recuperarlo tras la ejecución de algún fragmento de programa.

Los diagramas de tiempos

Los diagramas de tiempos, que presentamos seguidamente, contienen toda la información de cómo trabaja el Z-80 en sus diferentes fases a lo largo del tiempo de ejecución de un programa. En ellos se ve cómo evolucionan las distintas señales que hemos comentado en anteriores artículos.

En base a estas se diseñan los distintos periféricos que se conectan al Spectrum, y por supuesto, también nuestros montajes. En la figura número 1, se muestran los tres estados de ejecución por los que discurre constantemente el Z-80 en relación a la señal de reloj. El reloj, como ya se ha dicho, está fijado en el Spectrum a 3,5 Mhz; por tanto, su periodo de instrucción, denominados: M1, M2 y M3, emplea cuatro periodos de reloj, lo que equivale a 1,140 microsegundos. Estos tiempos son los que determinan la velocidad del Z-80, así si suponemos, por ejemplo, que una determinada instrucción requiere tres ciclos, ésta tardará en ejecutarse 3,42 microsegundos. Este tiempo parece minúsculo y despreciable desde nuestra perspectiva de movimientos humanos,pero resulta enorme a nivel electrónico. Esto se comprueba cuando una sucesión de instrucciones va acumulando microsegundos dando de resultas un tiempo notable en la ejecución de una rutina.

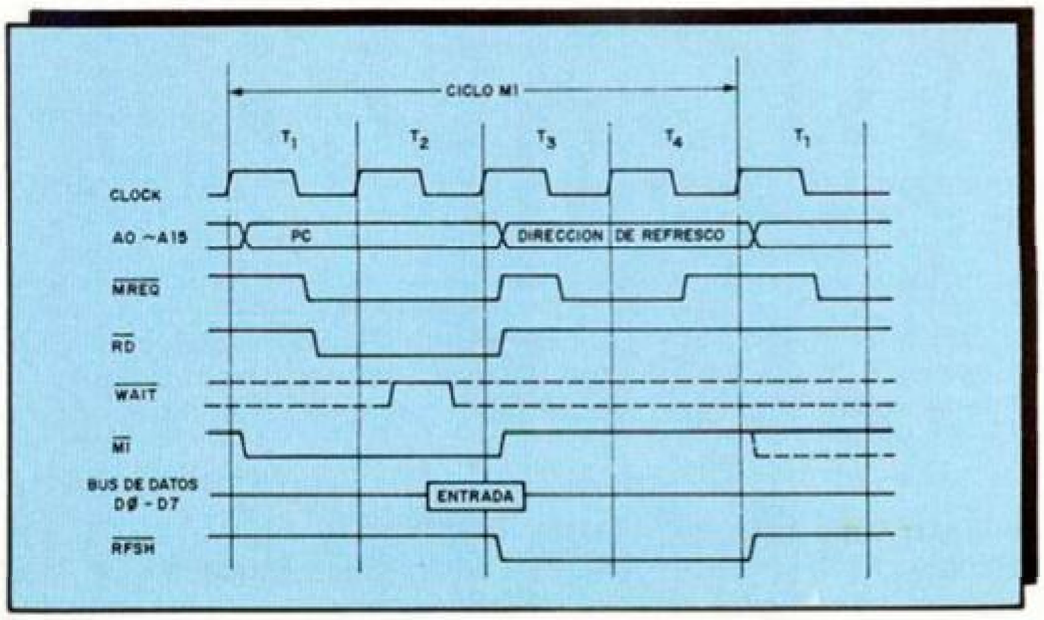


Fig. 2. Diagrama de estados lógicos en el momento de tomar el byte de código de una instrucción. En el dibujo se muestra la evolución de las señales del Z-80 que intervienen en este proceso.

En toda instrucción, el primer ciclo lo emplea el Spectrum para hacer la búsqueda del código de instrucción y decodificarlo, el segundo y tercer ciclo se suelen emplear en el acceso a memoria para la recogida del resto de los bytes que componen la instrucción. En la figura número 2, se representa el estado de las señales en que el Z-80 accede a la memoria para leer un byte. Si éste es el código de una instrucción, se genera durante dos periodos de reloj la señal M1.

pretende la ralentización de los buses para el uso de memorias o periféricos lentos.

En el momento cumbre de toda transparencia se habilita el bus de datos cuyo movimiento es, en esencia, lo que se pretende en cada ciclo de acceso a memoria o puertos.

Las señales RD y WR son determinantes en sus ciclos respectivos de lectura o escritura. Como se observa en las distintas figuras estas señales son

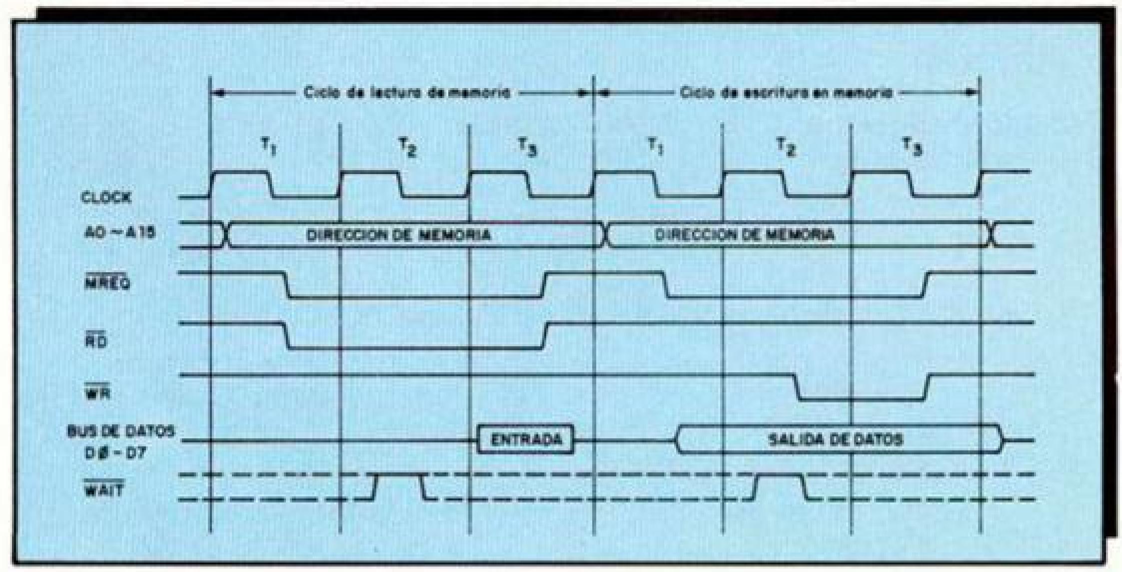


Fig. 3. Diagrama de estados lógicos en el momento de leer o escribir en memoria el Z-80.

En todo momento se han de alternar el ciclo de acceso o memoria y el de refresco de la misma. Obsérvese cómo en el acceso intervienen las señales MREQ y RD si es lectura, WD si es escritura (ver figura número 3).

La señal RFSH se produce durante el tiempo de refresco de las memorias RAM dinámicas. La señal WAIT que aparece en los diagramas es opcional y sólo se genera desde el exterior si se

activas en su estado bajo como la mayoría de las que intervienen en el Z-80. El estado alto es de reposo. En la figura número 5 se muestra el estado de las señales para el acceso a puertos tanto para escribir como para leer. Es obvio que RD y WR no pueden darse al tiempo aunque así represente en la figura, en donde se han comprendido sinópticamente ambos ciclos.

desplazamiento "d", el cual puede adquirir los valores desde −128 a +127. Deja el resultado en el registro acumulador.

CODIGO DE MAQUINA:

1 1 1 1 1 1 0 1	FDh
1 0 0 0 0 1 1 0	86h
< —— d —— >	

INDICADORES DE CONDICION A LOS QUE AFECTA:

S : pone 1 - si el resultado es negativo
pone 0 - en cualquier otro caso
Z : pone 1 - si el resultado es cero
pone 0 - en cualquier otro caso
H : pone 1 - si hay acarreo desde el bit 3
pone 0 - en cualquier otro caso
N : pone 0 - siempre
C : pone 1 - si hay acarreo desde el bit 7
pone 0 - en cualquier otro caso
P/V : pone 1 - si hay desbordamiento (overflow)
pone 0 - en cualquier otro caso

CICLOS DE MEMORIA:

5

CICLOS DE RELOJ:

19

EJEMPLO:

ADD A, (IY−6)

En esta ocasión vamos a usar, de nuevo, direccionamiento indexado para acceder al operando. Los dos números a sumar serán 7Fh y 03H, el resultado será 82h (F más 3 = 2 y nos llevamos una; 7 más 0 más 1 = 8). Como se ve, sumar en hexadecimal es lo mismo que hacerlo en decimal.

Valor del registro "IY"

(IY):	1 0 1 0 0 0 1 1	A3h
	0 1 1 1 1 0 0 0	78h

Valor de la posición de memoria A372h

(A372h):	0 1 1 1 1 1 1 1	7Fh

Valor del registro "A"

(A):	0 0 0 0 0 0 1 1	03h

Instrucción

	11111101	FDh
ADD A, (IY−6)	10000110	86h
	11111010	FAh

Valor del registro "A" después de la ejecución

(A):	1 0 0 0 0 0 1 0	82h

Indicadores de condición después de la ejecución

S	Z		H		P/V	N	C
1	0	x	1	x	1	0	0

Observe, que el indicador de signo (S) se activa por estar activo el bit 7, el tratar este número como negativo o no, dependerá del programador. El indicador P/V se activa por superar el máximo valor del octeto en complemento a dos (el bit de signo ha pasado de "0" a "1"). Finalmente, el indicador H está a "1" porque hubo acarreo desde el bit 3.

−.−.−.−

Las instrucciones de sumar, como su nombre indica, suman; pero con lo visto hasta el momento sólo suman un octeto. Por lo tanto se limita la suma al número 255, considerando todos positivos.

Este problema se soluciona con las instrucciones que se explican a continuación.

ADC (ADd with Carry), sumar con acarreo. Básicamente consiste en una suma binaria de dos octetos más el bit de acarreo. Esto quiere decir que si en una suma anterior el bit de acarreo está activo, "nos llevamos una", esa unidad hay que tenerla en cuenta en el octeto superior si existe.

Por ejemplo en una suma convencional en decimal.

7328+4256

al sumar las unidades 8 y 6 nos llevamos un 1 a las decenas; con las decenas y las centenas no hay acarreo, y de nuevo en las unidades de millar hay acarreo a las decenas de millar.

```
  1    1      acarreo
   7328
 + 4256
  -----
  11584
```

Visto esto, se entenderá fácilmente, que sumando octetos se acarrea 1 al octeto superior cuando se supera el valor decimal 255 (FFh). Ver FIGURA 6-1.

Esta es la manera de sumar, en binario, cantidades superiores a 255 decimal; usando las instrucciones que se describen a continuación.


```
LD HL,(#5CB2)
PUSH HL
POP BC
RET
```

Ahora, codificamos el programa, buscamos en las tablas el código de:

LD HL, (nn)

que resulta ser 2Ah (42), a continuación van los operandos B2h (178) y 5C (92). El código de:

PUSH HL

es E5h (229), y el de:

POP BC

es C1h (193). Por último, colocamos el código de RET: C9h (201). El programa completo, queda de la siguiente forma:

2A,B2,5C,E5,C1,C9

O, para quienes lo prefieran en decimal:

42,178,92,229,193,201

Ahora, sólo nos falta cargarlo en una línea REM de un programa en Basic. Nuestra rutina tiene 6 bytes, por lo que crearemos una línea REM con, por ejemplo, 6 asteriscos. Estos asteriscos serán sustituidos por los bytes del programa cuando éste se cargue.

El programa en Basic podría ser el siguiente:

```
   10 REM ******
   20 LET prog=PEEK 23635+256*PEE
K 23636
   30 FOR n=5 TO 10
   40 READ a: POKE prog+n,a
   50 NEXT n
   60 DATA 42,178,92,229,193,201
   70 PRINT USR (prog+5)
```

Hay muchos puntos sutiles en este programa que conviene analizar detenidamente; como dijimos antes, la línea **10** contiene el espacio donde se cargará nuestra rutina en C/M.

En la línea **20**, leemos la variable del Sistema PROG, para saber a partir de qué dirección de memoria está ubicado el programa Basic. Los dos primeros bytes de esta zona, constituyen el número de línea, los dos siguientes la longitud, y el quinto es el código de REM; a partir de ahí empiezan los asteriscos, que es donde deberemos cargar el código máquina, es decir, desde "prog+5" hasta "prog+**10**" tal y como se ve en las líneas **30**, **40** y **50**. La línea **60** contiene el programa en DATAs. Finalmente, la línea **70** ejecuta el programa desde la dirección "prog+5". En este caso, es imprescindible que el argumento de USR vaya entre paréntesis; es muy fácil omitir los paréntesis, olvidándose de que la función USR tiene una prioridad más alta que la suma.

Una vez ejecutado el programa, la línea **10** quedaría:

```
   10 REM *SIN \ RESTORE STR$ <>
```

Nuestro siguiente ejemplo es más vistoso, y algo más complejo. Vamos a dibujar una silueta en pantalla, y dado que la cosa va de pantalla, almacenaremos esta rutina en el archivo de presentación visual, con lo que veremos físicamente los bytes que la componen, en forma de pixels en la primera línea.

El objetivo del programa es dibujar en la casilla central, la silueta de un muñeco, como si se tratara de un UDG. Dado que sólo podemos utilizar instrucciones de carga, el programa resulta considerablemente más largo de lo que es normal para trabajar con la pantalla.

En sucesivos ejemplos de capítulos más avanzados, iremos viendo otras formas más sencillas de imprimir en pantalla; y veremos también, cómo la peculiar manera en que está organizado el archivo de pantalla, que tan incómoda se hacía en Basic, resulta una gran ventaja cuando se trabaja en código máquina.

La forma más sencilla de imprimir un gráfico en pantalla, es almacenar en las ocho direcciones que componen una casilla, los ocho números que definen ese gráfico. En la FIGURA 5-10, vemos las direcciones de las posiciones de memoria correspondientes a la casilla central de la pantalla, así como los datos que vamos a almacenar en esas posiciones, para visualizar nuestro muñeco.

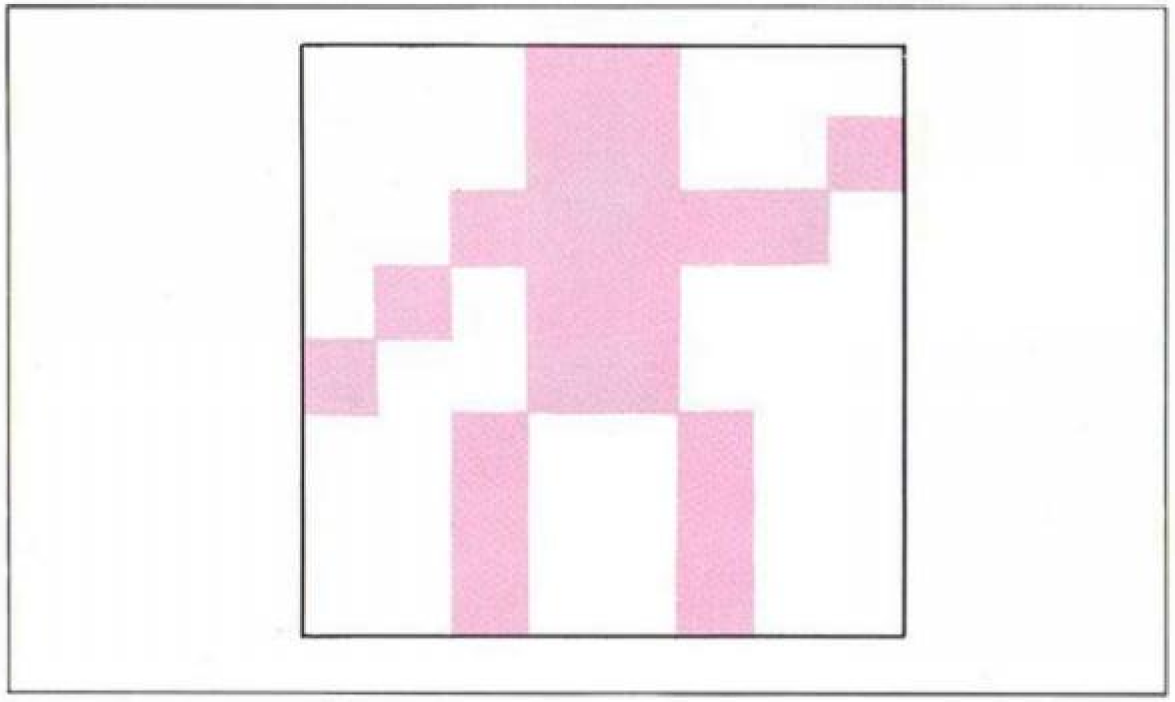


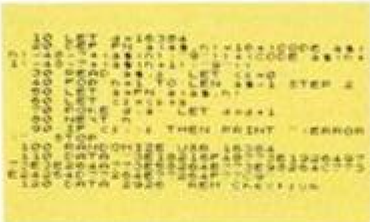
Fig. 5-10: Datos y direcciones para crear un muñeco en pantalla.

El método general que vamos a utilizar, es cargar el registro "A" con el dato a almacenar, el registro "HL" con la dirección, y almacenar el dato "A" en la dirección apuntada por "HL". Como todavía no hemos aprendido a hacer bucles, tendremos que repetir esta secuencia 8 veces, si bien, las veces sucesivas será suficiente con que modifiquemos el valor del registro "H", ya que el del "L" permanece constante. El listado sería el siguiente:

```
LD A,#18        3E,18
LD HL,#486F     21,6F,48
LD (HL),A       77
LD A,#19        3E,19
LD H,#49        26,49
LD (HL),A       77
LD A,#3E        3E,3E
LD H,#4A        26,4A
LD (HL),A       77
LD A,#58        3E,58
LD H,#4B        26,4B
LD (HL),A       77
LD A,#98        3E,98
LD H,#4C        26,4C
LD (HL),A       77
LD A,#24        3E,24
LD H,#4D        26,4D
LD (HL),A       77
LD H,#4E        26,4E
LD (HL),A       77
LD H,#4F        26,4F
LD (HL),A       77
RET             C9
```

Obsèrvese que, dado que las tres últimas líneas del dibujo son iguales, no ha sido necesario cargar el registro "A" más que 6 veces.

A la derecha del listado Assembler, está el código máquina de cada una de las instrucciones. En este caso, vamos a utilizar una técnica más refinada para cargar el programa; escribiremos el código máquina en hexadecimal sobre una línea DATA del Basic, y utilizaremos una suma de control para detectar posibles errores. El programa en Basic sería el siguiente:



Primero fijamos la dirección de carga al inicio del archivo de presentación visual. En la línea 20, definimos una función que nos ayude a convertir de hexadecimal a decimal, la línea 30 lee el código máquina en la variable "a$", la suma de comprobación en la viariable "s" y pone a cero

S	Z		H		P/V	N	C
0	0	x	0	x	0	0	1

Observe, que ha habido acarreo desde el bit 7

ADD A, (IX+d)

OBJETO:

Suma el registro acumulador "A" con el octeto de la posición de memoria direccionada por el valor que resulta de: añadir al contenido del registro indice "IX" el entero de desplazamiento "d", el cual puede adquirir los valores desde −128 a +127. Deja el resultado en el registro acumulador.

CODIGO MAQUINA:

1 1 0 1 1 1 0 1	DDh
1 0 0 0 0 1 1 0	86h
<—— d ——>	

INDICADORES DE CONDICION A LOS QUE AFECTA:

- S ; pone 1 - si el resultado es negativo; pone 0 - en cualquier otro caso
- Z ; pone 1 - si el resultado es cero; pone 0 - en cualquier otro caso
- H ; pone 1 - si hay acarreo desde el bit 3; pone 0 - en cualquier otro caso
- N ; pone 0 - siempre
- C ; pone 1 - si hay acarreo desde el bit 7; pone 0 - en cualquier otro caso
- P/V ; pone 1 - si hay desbordamiento (overflow); pone 0 - en cualquier otro caso

CICLOS DE MEMORIA:

5

CICLOS DE RELOJ:

19

EJEMPLO:

ADD A,(IX+7)

Suponemos que el registro "A" contiene el número 80h (128) y vamos a sumarle el contenido de la posición de memoria 9318h que suponemos que es también 80 (128). Para acceder a esta posición utilizamos direccionamiento indexado. El resultado deberá ser 256, es decir, 100h; pero como este número excede la capacidad del acumulador, obtendremos "00h" en el acumulador y "1" en el indicador de acarreo.

Esto ocurrirá con frecuencia. Cuando el resultado de una suma sea mayor de 255 (FFh), nos aparecerá el flag de acarreo a "1", y el acumulador contendrá ese número menos 256, para los matemáticos, lo que obtenemos en el acumulador es el "módulo 256" del resultado de la suma. Cuando, después de una suma, el indicador de acarreo sea "1", podemos saber el verdadero resultado si sumamos 256 al contenido del acumulador. Veamos ahora el ejemplo.

Valor del registro "IX":

(IX)	1 0 0 1 0 0 1 1	93h
	0 0 0 1 0 0 0 1	11h

Valor de la posición de memoria 9318h:

(9318h)	1 0 0 0 0 0 0 0	80h

Valor del registro "A"

(A)	1 0 0 0 0 0 0 0	80h

Instrucción

ADD A, (IX+7)	11011101	DDh
	10000110	86h
	00000111	07h

Valor del registro "A" después de la ejecución

(A)	0 0 0 0 0 0 0 0	00h

Indicadores de condición después de la ejecución

S	Z		H		P/V	N	C
0	1	x	0	x	1	0	1

Obsèrvese que se nos han puesto a "1" los indicadores de acarreo, cero y rebosamiento. El de acarreo, porque el resultado es mayor de 255; el de cero, porque el acumulador contiene "cero"; y el de rebosamiento, porque el bit 7 ha pasado de ser "1" a ser "0", lo que se interpreta como un cambio de signo; en este caso, el signo no nos interesa porque el número no puede ser negativo, por lo que, simplemente, ignoramos el indicador "P/V".

ADD A, (IY+d)

OBJETO:

Suma el registro acumulador "A" con el octeto de la posición de memoria direccionada por el valor que resulta de: añadir al contenido del registro indice "IY" el entero de

Indicadores de condición después de la ejecución:

S	Z		H		P/V	N	C
0	0	x	1	x	0	0	0

Observe, que hubo acarreo desde el bit 3.

ADD A,n

OBJETO:

Suma el registro acumulador "A" con el número entero de 8 bits "n", dejando el resultado en el registro acumulador.

CODIGO DE MAQUINA:

11000110	C6h
<— n —>	

INDICADORES DE CONDICION A LOS QUE AFECTA:

S ; pone 1 - si el resultado es negativo
pone 0 - en cualquier otro caso
Z ; pone 1 - si el resultado es cero
pone 0 - en cualquier otro caso
H ; pone 1 - si hay acarreo desde el bit 3
pone 0 - en cualquier otro caso
N ; pone 0 - siempre
C ; pone 1 - si hay acarreo desde el bit 7
pone 0 - en cualquier otro caso
P/V ; pone 1 - si hay desbordamiento (overflow)
pone 0 - en cualquier otro caso

CICLOS DE MEMORIA:

2

CICLOS DE RELOJ:

7

EJEMPLO:

ADD A,24

Valor del registro "A":

(A): 00100110 26h

Instrucción

ADD A,24	11000110	C6h
	00011000	18h

Valor del registro "A" después de la ejecución:

(A): 00111110 3Eh

Indicadores de condición después de la ejecución:

S	Z		H		P/V	N	C
0	0	x	0	x	0	0	0

ADD A,(HL)

OBJETO:

Suma el registro acumulador "A" con el octeto de la posición de memoria direccionada por el contenido del par de registros "HL", y deja el resultado en el registro acumulador.

CODIGO DE MAQUINA:

10000110 86 h

INDICADORES DE CONDICION A LOS QUE AFECTA:

S ; pone 1 - si el resultado es negativo
pone 0 - en cualquier otro caso
Z ; pone 1 - si el resultado es cero
pone 0 - en cualquier otro caso
H ; pone 1 - si hay acarreo desde el bit 3
pone 0 - en cualquier otro caso
N ; pone 0 - siempre
C ; pone 1 - si hay acarreo desde el bit 7
pone 0 - en cualquier otro caso
P/V ; pone 1 - si hay desbordamiento (overflow)
pone 0 - en cualquier otro caso

CICLOS DE MEMORIA:

2

CICLOS DE RELOJ:

7

EJEMPLO:

ADD. A,(HL)

Valor del par de registros "HL":

(H):	01001100	4Ch
(L):	11110011	F3h

Valor de la posición de memoria 4CF3h:

(4CF3h): 10011100 9Ch

Valor del registro "A":

(A): 10110000 B0h

Instrucción:

ADD A,(HL) 10000110 86h

Valor del registro "A" después de la ejecución:

(A): 01001100 46h

Indicadores de condición después de la ejecución:

DATOS	DIRECCIONES		
1 8	4 8 6 F	9 8	4 C 6 F
1 9	4 9 6 F	2 4	4 D 6 F
5 E	4 A 6 F	2 4	4 E 6 F
5 8	4 B 6 F	2 4	4 F 6 F

el acumulador de checksum "cs". Las líneas 40 a 80 van pasando los códigos a decimal (línea 50), acumulándolos en el checksum (línea 60) y metiéndolos en sucesivas direcciones de memoria (línea 70).

La línea 90 comprueba que el valor acumulado en checksum sea igual a la suma de comprobación, y détiene el programa en caso contrario. Finalmente, la línea 100 ejecuta nuestra rutina en código máquina. La línea 110 contiene el código máquina en hexadecimal, y la 120 la suma de todos los bytes en decimal, que se utiliza como suma de comprobación.

Cuando se ejecute el programa, en la pantalla del ordenador tiene que aparecer algo similar a lo que se ve en la FIGURA 5-11.

En el siguiente capítulo, veremos las instrucciones que nos permiten realizar operaciones aritméticas y lógicas sobre los registros del microprocesador. Antes de ello, le recomendamos al lector que intente resolver los siguientes ejercicios, que le ayudarán a afianzar conocimientos.

EJERCICIOS

1.- ¿Que valor retornará el siguiente programa en el registro "BC"?. ¿Que valor tendremos en la variable del sistema "SEED" despues de ejecutarlo?:

```
        LD HL,#4BFF
        LD BC,#1100
        LD H,C
        LD (SEED),BC
        PUSH HL
        POP BC
        RET
SEED    EQU #5C76
```

(Puede encontrar la solución por usted mismo, codificando el programa y ejecutándolo en el ordenador, para comprobar si la solución que ha dado es correcta).

2.- Codificar (ensamblar) el siguiente programa:

```
        LD BC,#1234
        LD A,(BC)
        LD (IX+7),A
        PUSH AF
        POP BC
        RET
```

3.- Escribir cuatro cargadores en Basic, cada uno de los cuales almacenen el programa anterior en uno de los siguientes lugares:

EN EL BUFFER DE IMPRESORA
ENCIMA DE RAMTOP
EN UNA LINEA REM
EN LA PANTALLA

INSTRUCCIONES ARITMETICAS Y LOGICAS

El microprocesador Z-80 dispone de una unidad aritmética-lógica que le permite realizar una serie de operaciones, tanto aritméticas, como lógicas. Las aritméticas incluyen la suma y resta con o sin acarreo, incremento y decremento de un registro, comparaciones, ajuste decimal, complemento y negación. Las lógicas incluyen las operaciones que se realizan con los operadores "AND", "OR" y "XOR".

Antes de adentrarnos en el estudio de las instrucciones concretas, daremos una serie de definiciones útiles:

SUMA SIN ACARREO:

Consiste en sumar al contenido del registro "A" un número y obtener el resultado en el registro "A". El indicador de acarreo no se tiene en cuenta para esta operación. Su esquema sería:

A ← A+n

SUMA CON ACARREO:

Exactamente igual que la anterior, pero se suma también el indicador de acarreo del registro "F". De esta forma, se puede incluir en la suma el acarreo procedente de una suma anterior. Su esquema sería:

A ← A+n+CF

RESTA SIN ACARREO:

Consiste en restar un número del contenido del registro "A", y obtener el resultado en este mismo registro. El indicador de acarreo no interviene en la operación. Se consideran números negativos los superiores a 127 (7Fh) de la forma que se explicó en el capítulo relativo a los sistemas de numeración; es decir, el número 255 (FFh) se considera "−1", el 254 (FEh) se considera "−2" y así sucesivamente, hasta 128 (80) que se considera "−128". El paso de 127 a 128 o viceversa se indica poniendo a "1" el flag de "overflow" (P/V) del registro "F". Su esquema sería:

A ←A−n

RESTA CON ACARREO:

Igual que el anterior, salvo que también se resta el indicador de acarreo (CF) del registro "F". Su esquema sería:

A ← A−n−CF

INCREMENTO:

Consiste en sumar uno al contenido de un registro que se especifica en la instrucción. Su esquema es:

R ← R+1

Donde "R" representa un registro cualquiera de 8 a 16 bits. Si se trata de un registro doble (de 16 bits) se incrementa el registro de orden bajo (por ejemplo, en el "BC" se incrementa "C"), y si ello hace que éste pase a valer "0", se incrementa también el orden alto.

DECREMENTO:

Es la inversa de la anterior, consiste en restar uno al contenido de un registro. Su esquema es:

R ← R−1

Si se trata de un registro doble, se decrementa el de orden bajo y, si esto hace que pase a valer 255 (FFh), se decrementa también el de orden alto.

Si el registro incrementado o decrementado es de 8 bits, resultan afectados los indicadores del registro "F".

COMPARACIONES:

Estas instrucciones permiten comparar el contenido del acumulador con un número. Para ello, se resta el número del contenido del acumulador, pero el resultado no se almacena en ninguna parte, simplemente, se alteran determinados flags del registro "F", lo que nos indica si el número era menor, igual o mayor que el contenido del acumulador. Si era igual, se pone a "1" el flag "Z" (indicador de "cero"). Si el número era mayor, se pone a "1" el flag "S" (indicador de "signo").

AJUSTE DECIMAL:

Esta instrucción realiza un ajuste del contenido del acumulador para que, en vez de estar comprendido entre "00h" y "FFh", lo esté entre "00h" y "99h". Si se produce acarreo, se indica mediante el flag correspondiente. Para realizar esta operación se toma en cuenta el estado de los indicadores de "acarreo" (C) y "semi-acarreo" (H). Su finalidad es la de permitir realizar operaciones en "BCD" (Decimal Codificado en Binario).

COMPLEMENTO:

Consiste en realizar un "completo a 1" del acumulador, es decir, cambiar los "unos" por "ceros" y los "ceros" por "unos".

NEGACION:

Consiste en realizar un "complemento a 2" del acumulador, es decir, realizar un "complemento 1" y, luego, sumarle "1". Lo que se obtiene es el "negativo" del número que teníamos en el acumulador. El efecto es el mismo que si restáramos el acumulador de "cero", es decir:

A ← 0−A

EL FLAG DE ACARREO:

Existen dos instrucciones que afectan al indicador de acarreo del registro "F", es posible ponerlo a "1" o "complementarlo" (ponerlo a "1" si era "0" y viceversa). No se ha previsto una instrucción para poner a "0" el flag de acarreo, dado que esto se puede conseguir haciendo un "AND" o un "OR" del acumulador consigo mismo.

Veamos ya las instrucciones:

Grupo de instrucciones aritméticas para 8 bits

En este grupo de instrucciones los registros usados se indican con "r" según el siguiente código:

"r"	código
A	111
B	000
C	001
D	010
E	011
H	100
L	101

ADD, «sumar» en inglés: La función básica de esta instrucción es sumar sobre el registro acumulador el valor indicado por el operando. Ejecuta una suma binaria de ambos datos y no altera el contenido del operando.

ADD A,r

OBJETO:

Suma el registro acumulador "A" con el registro indicado por "r", dejando el resultado en el registro acumulador.

CODIGO MAQUINA:

1 0 0 0 0 ← r

INDICADORES DE CONDICION A LOS QUE AFECTA:

S ; pone 1 - si el resultado es negativo
pone 0 - en cualquier otro caso
Z ; pone 1 - si el resultado es cero
pone 0 - en cualquier otro caso
H ; pone 1 - si hay acarreo desde el bit 3
pone 0 - en cualquier otro caso
N ; pone 0 - siempre
C ; pone 1 - si hay acarreo desde el bit 7
pone 0 - en cualquier otro caso
P/V ; pone 1 - si hay desbordamiento (overflow)
pone 0 - en cualquier otro caso

NOTA: Se entiende que hay acarreo desde el bit 3 cuando éste pasa de ser "1" a ser "0". Se entiende que hay desbordamiento si el resultado pasa de ser "positivo" a ser "negativo" o viceversa. Estas observaciones son válidas para todas las operaciones aritméticas.

CICLOS DE MEMORIA:

1

CICLOS DE RELOJ:

4

EJEMPLO:

ADD A,B

Valor del registro "A":

(A): 0 0 1 0 1 0 0 1 29h

Valor del registro "B":

(B): 0 1 0 0 1 0 1 0 4Ah

Instrucción:

ADD A,B: 1 0 0 0 0 0 0 0 80h

Valor del registro "A" después de la ejecución:

(A): 0 1 1 1 0 0 1 1 73h

El valor del registro "B" después de la ejecución no varía.



COMO PASAR PROGRAMAS DE CINTA A MICRODRIVE

Luis E. JUAN

La semana pasada enumerábamos las distintas posibilidades o métodos de conversión que existen para traspasar programas de cinta a Microdrive. Ahora vamos a entrar en materia con procedimientos detallados y algunos ejemplos prácticos.

PROGRAMAS INTEGRAMENTE EN BASIC

a) Son programas cuya cabecera es del tipo:

```
Nombre:   PROGBASIC
Tipo:     BASIC
Origen:   L10
Longitud: 1234 bytes
```

Para su paso a microdrive basta con cargarlos mediante MERGE "" y grabarlos con un SAVE • "m";1; "PROGBASIC" LINE 1Ø (en el ejemplo).

A esta categoría pertenecen también los llamados «cargadores», que sirven para cargar desde un programa BASIC todas las secciones de código máquina, etc.), con o sin cabecera, que constituyen normalmente el «cuerpo» del programa. Previamente a su almacenamiento en microdrive, deberemos hacer en ellos ciertos cambios, según se verá en el apartado correspondiente.

Si el programa no admite el "MERGE", será preciso cargarlo fabricando previamente una cabecera falsa (para obviar el auto-RUN). Para ello, se introduce el Programa 1, grabándolo para futuros usos. Se hace un RUN y se contesta (en el ejemplo enterior): Ø (para seleccionar BASIC) y 1234 (longitud). Se pone una cassette virgen y el magnetófono en grabación, pulsando a continuación una tecla. El programa grabará únicamente una cabecera falsa, que emplearemos del modo siguiente: Hacemos avanzar la cinta original hasta la pequeña pausa entre la cabecera original y el programa. Se extrae la cinta original. Se hace un NEW y un LOAD "", cargando entonces la cabecera falsa. A continuación, sin tocar el SPECTRUM, ponemos la cinta original y pulsamos PLAY en el magnetófono. El bloque del programa cargará normalmente como en un "MERGE". Este procedimiento también es válido, como se verá, para cargar programas sin cabecera.

PROGRAMAS BASIC CON CODIGO MAQUINA EN LINEAS REM

b) Estos programas tienen una cabecera como los (a), pero al intentar listarlos aparecen sentencias del tipo 1 REM... seguidas de códigos sin sentido. Desgraciadamente, casi todos los programas de esta categoría son totalmente incompatibles con el microdrive, por lo que si no funcionan al ser tratados con el procedimiento (a) sólo habrá dos opciones: armarse de paciencia y desensamblador y modificar el código o (generalmente) resignarse a dejarlos en cassette.

PANTALLAS DE PRESENTACION

C) Las pantallas de presentación tienen una cabecera del tipo:

```
Nombre:   PANTALLA
Tipo:     Bytes
Origen:   16384 (puede ser otro)
Longitud: 6912 bytes (caracteristica)
```

Se recomienda cargarlas fuera de la memoria de pantalla para evitar la pérdida de las dos líneas inferiores. Un ejemplo del método sería:

```
CLEAR 24999: LOAD ""CODE 25000:
SAVE •"m";1;"nombre"CODE 25000,6
912
```

En el programa cargador deberemos poner:

```
LOAD •"m";1;"nombre"SCREEN$
```

CODIGO MAQUINA CON ORIGEN SUPERIOR A 25ØØØ

d) Son secciones de código máquina del tipo:

```
Nombre:   CODIGO1
Tipo:     Bytes
Origen:   32768
Longitud: 24576
```

Si el cargador BASIC que precede al código máquina es suficientemente corto, pertenecerán también a esta categoría los bloques de código con origen superior a 248ØØ.

Para pasar a microdrive el código del ejemplo, se haría:

```
CLEAR 32767: LOAD ""CODE 32768:
SAVE •"m";1;"CODIGO1"CODE 32768,
28672
```

En el programa cargador se pondría:

```
LOAD •"m";1;"CODIGO1"CODE 32768,
28672
```

NOTA IMPORTANTE.—Siempre que se vaya a cargar código máquina es buena práctica hacer previamente un CLEAR origen —1 para evitar que en algún caso se altere la dirección de retorno de error, con el consiguiente bloqueo del sistema.

CODIGO MAQUINA CON ORIGEN INFERIOR A 25ØØØ Y LONGITUD MENOR DE 4Ø5ØØ

e) Un ejemplo de esta categoría sería:

```
Nombre:   CODIGO2
Tipo:     Bytes
Origen:   24576
Longitud: 32768
```

En este caso, el código cargado en su dirección original entraría en conflicto con la zona reservada al BASIC o, peor aún, con el área de trabajo del microdrive, causando pérdida de control del sistema. Recordemos que en las operaciones con microdrive la primera posición de memoria disponible es 2444Ø y el RAMTOP en el ejemplo debería estar situado en 24575, con lo que habría sólo 135 bytes libres para contener toda la información desde la dirección PROG (entre otras cosas, el cargador BASIC). La solución no es cargar el código desplazado, grabarlo así en el microdrive y cargarlo luego en la dirección original, pues el efecto es el mismo.

El método obligado es el siguiente:

En primer lugar, es preciso averiguar cuál es la dirección de ejecución del código. Esto ha de hacerse examinando el cargador BASIC que necesariamente le precede en la cassette. La llamada puede aparecer en la forma: RANDOMIZE USR XXXXX o bien PRINT USR XXXXX o incluso LET variable= USR XXXXX. Esta "XXXXX" es la dirección de ejecución del código máquina. Supongamos en el ejemplo la 26ØØØ.

Seguidamente, se introduce el Programa 2, grabándolo para posteriores usos. Se carga entonces el código desplazado por encima de la dirección 25ØØØ. En el ejemplo anterior podría hacerse:

```
CLEAR 29999: LOAD ""CODE 30000
```

Haciendo un RUN, el Programa 2 nos pedirá:

```
Origen actual= 30000
Longitud= 32768
Origen final= 24576
Dirección de
   ejecución=26000
```

El programa añade automáticamente una pequeña rutina de 14 bytes a continuación del bloque principal, que al ser llamada traslada el código a su verdadero origen y transfiere el control a la dirección de ejecución. Asimismo, nos informa de los parámetros a utilizar en la grabación y posterior ejecución.

Siguiendo con el ejemplo, se grabaría en microdrive con:

```
SAVE *"m";1;"CODIGO2"CODE 30000,
32782
```

y se ejecutaría desde el cargador con RANDOMIZE USR 62768.

Los programas pertenecientes a esta categoría o a la siguiente cuyo origen sea inferior a 23700 incluyen las variables del sistema (direcciones 23552 a 23733), lo cual indica que fueron grabados en ejecución BASIC como código máquina. La dirección de ejecución a suministrar al Programa 2 será "0", con lo cual tras la reubicación del código se devuelve el control a dicho BASIC, que será quien ejecute el código.

CODIGO MAQUINA CON ORIGEN INFERIOR A 25000 Y LONGITUD MAYOR DE 40500

f) Es el caso de los programas del tipo:

```
Nombre:   CODIGO3
Tipo:     Bytes
Origen:   23552
Longitud: 41984
```

En este caso no hay sitio en memoria para cargar el código desplazado por encima del área de trabajo del microdrive. De hecho, en el ejemplo el código se superpone incluso a las variables del sistema.

El procedimiento a seguir es bastante más complejo y tedioso que en los casos anteriores, teniendo como filosofía la de dividir el bloque en 2 secciones: la parte alta, por encima de la dirección 25000, se procesa como en el caso (d) y la baja se procesa como en el (e), pero utilizando la zona de memoria de pantalla como almacenamiento temporal. Se expone a continuación el método particularizado para el ejemplo, que es fácilmente adaptable a cualquier caso.

En primer lugar, se carga el código despalzado hacia arriba 2000 bytes:

```
CLEAR 25551: LOAD "CODIGO3"CODE
25552
```

Dado que 25552 + 41984 > 65536, los 2000 bytes superiores se pierden, ya que el sistema operativo los intenta cargar en ROM (después de la dirección 65535, el Z80 pasa a la 0).

Se graba en microdrive el código desde 2000 bytes más arriba del nuevo origen:

```
SAVE *"m";1;"MEDIO"CODE 27552,37
984
```

Para recuperar la parte alta del programa haremos:

```
LOAD "CODIGO3"CODE 41936
```

Este origen es tal que los últimos 2000 bytes quedan almacenados en la parte baja de la memoria de pantalla. (41936 = 65536 + 16384 + 2000 − 41984).

Con precaución para no borrar la pantalla, haremos:

```
SAVE *"m";1;"ALTO"CODE 16384,200
0
```

Seguidamente, haremos un RESET (RANDOMIZE USR 0) e introduciremos el Programa 2, cargando a continuación de nuevo el código con un: LOAD "CODIGO3" CODE 16384 y deteniendo la cinta cuando se haya llenado completamente el tercio superior de la pantalla (2048 bytes).

Hacemos correr el Programa 2 con un GO TO 10 (no con un RUN, pues borraría la pantalla). Los parámetros a suministrar son:

```
Origen actual= 16384
Longitud= 2000
Origen final= 23552
Dirección de ejecución
= 0 (ver párrafo final del
procedimiento (e))
```

Se graba en microdrive con:

```
SAVE *"m";1;"CODIGO3"CODE 16384,
2014
```

Ahora reunimos los bloques "MEDIO" y "ALTO", ubicándolos en su dirección de ejecución.

```
CLEAR 25551: LOAD *"m";1;
"MEDIO" CODE 25552: LOAD
*"m";1;"ALTO"CODE 63536
```

y los grabamos con un:

```
SAVE *"m";1;"CODIGO3H"CODE
25552,39984
```

Los productos intermedios ("MEDIO" y "ALTO" pueden ser eliminados con los ERASEs correspondientes).

Finalmente, escribimos y almacenamos el siguiente cargador:

```
10 CLEAR 25551
20 LOAD *"m";1;"CODIGO3H"CODE
25552,39984
30 LOAD *"m";1;"CODIGO3L"CODE
16384,2014
40 RANDOMIZE USR 16384
```

Si el código objeto de esta categoría tuviera un origen superior o igual a 24000, el método se simplifica, ya que es posible cargar desde cassette todo el código en memoria de una vez (CLEAR 23999 : LOAD "" CODE) y separarlo en 2 bloques de forma directa. En este caso, al igual que en el (e) es preciso averiguar previamente la dirección de ejecución para suministrarla al Programa 2.

Si, por el contrario, el bloque de código máquina contiene incluso la zona de pantalla, será preciso prescindir de ella, eliminándola a base de cargar el código desplazado, de modo análogo al del ejemplo.

NOTA.— El umbral de 25000 que hemos establecido en las categorías (d), (e) y (f) puede ser superior dependiendo de la longitud del BASIC que aconpañe al código máquina. Como regla general aproximada, el Umbral = 24442 + Longitud del programa BASIC.

PROGRAMA 1

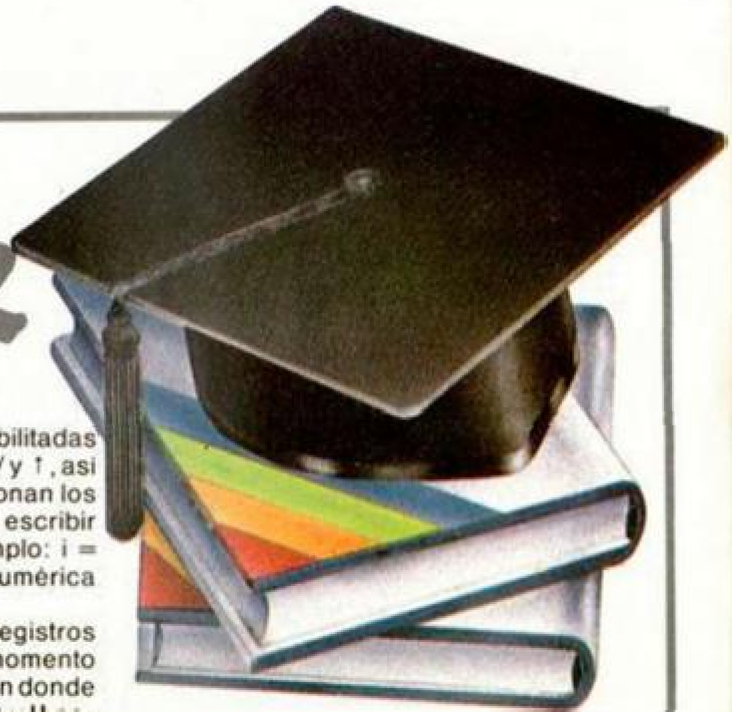
```
 10 INPUT "0= BASIC  3= CODIGO
MAQUINA ";X
 20 INPUT "LONGITUD= ";I
 30 PRINT AT 11,0;("BASIC" AND
NOT X)+("CODIGO MAQUINA" AND X);
", LONGITUD= ";I
 40 LET H=INT (I/256): LET L=I-
256*H
 50 FOR I=30000 TO 30030: READ
N: POKE I,N: NEXT I
 60 PRINT #0;"ARRANQUE CINTA Y
PULSE UNA TECLA": PAUSE 0
 70 PAUSE 0: RANDOMIZE USR 3001
8
 80 DATA X,70,65,76,83,65,67,65
,66,69,90,L,H,0,128,0,0,0,221,33
,48,117,17,17,0,175,55,205,194,4
,201
```

PROGRAMA 2

```
 10 DEF FN L(X)=X-256*INT (X/25
6): DEF FN H(X)=INT (X/256)
 20 INPUT "Origen actual= ";oa
 30 INPUT "Longitud= ";lo
 40 INPUT "Origen final= ";of
 50 INPUT "Direccion ejecucion="
;de
 60 LET p=oa+lo
 70 FOR i=p TO p+13: READ j: PO
KE i,j: NEXT i
 80 PRINT AT 16,0;"Grabar con S
AVE CODE ";oa;",";lo+14
 90 PRINT : PRINT "Ejecutar con
USR ";p
 100 DATA 33,FN L(oa),FN H(oa),1
,FN L(lo),FN H(lo),17,FN L(of),F
N H(of),237,176,195+6*(de=0),FN
L(de),FN H(de)
```



Profesor particular



NUMEROS COMPLEJOS

En esta ocasión os ofrecemos un programa con el que podéis operar con números complejos y representarlos en la pantalla. El programa permite calcular cualquier expresión en la que intervengan sumas, restas, multiplicaciones, divisiones y potencias con argumentos reales o complejos. Asimismo, podéis emplear paréntesis y podéis introducir los números en forma binomial o polar. A continuación, os explicamos el manejo que, como veréis, es muy sencillo para las posibilidades que ofrece:

En primer lugar, al comenzar se requieren dos cantidades: FIX es el número de decimales con el que queréis que se presenten en pantalla los resultados (recomendamos 2), y Xmax es el fondo de escala de la pantalla de representación de los afijos.

Cuando hayáis intoducido ambos datos, el programa estará listo para calcular la expresión que queráis. Las teclas habilitadas son: los números, paréntesis, + — • / y ↑, así como «TO» y «THEN», que proporcionan los signos L y º respectivamente para escribir los números en forma polar (Ejemplo: i = 1L90º). Para obtener el resultado numérica y gráficamente, pulsad ENTER.

El programa prevee además, 5 registros para guardar los datos. En cada momento se encuentra habilitado el registro en donde figura el asterisco. Con las teclas ⇑ y ⇓ podéis pasar de un registro a otro. Con la tecla M memorizáis en ese registro el resultado de la operación. Con la tecla C cambiáis la forma de presentación en el registro de binomial a polar y viceversa. La tecla R representa el registro en la pantalla gráfica. Esto es útil puesto que, aunque los afijos se van representando directamente, tenéis la posibilidad de borrar la pantalla con la tecla B.

Por último, las teclas E y F cambian la escala (Xmax) y el FIX respectivamente. El cambio de escala borra y vuelve a representar todo lo que hubiera en la nueva escala. Para efectuar otro cálculo pulsad enter y el display estará listo de nuevo.

Finalmente un detalle. Si la representación del afijo se lleva a cabo en color rojo, esto indica que nos hemos salido de la escala, por lo que éste no puede visualizarse completamente. Cambiando la escala, asunto arreglado.

```
10 BORDER 5: PAPER 5: CLS
20 FOR i=1 TO 7: READ a,b,c,dd: GO SUB 100: NEXT i
25 FOR i=USR "a" TO USR "a"+15: READ a: POKE i,a: NEXT i
30 DATA 0,1,0,31,4,4,2,11,7,7,2,11,10,10,2,11,13,13,2,11,16,16,2,11,4,21,14,31,64,64,64,64,64,64,127,127,0,24,36,36,24,0,0,0
95 GO TO 130
100 FOR i=a TO b: FOR j=c TO dd
110 PRINT AT i,j; PAPER 2;" "
120 NEXT j: NEXT i: RETURN
130 LET d=0: DIM z(10,4): DIM r(5,5)
140 PAPER 7: PRINT AT 4,0;"u";AT 7,0;"v";AT 10,0;"x";AT 13,0;"y";AT 16,0;"z"
150 INK 6: PLOT 183,0: DRAW 0,144: PLOT 255,72: DRAW -144,0: INK 1: CIRCLE 183,72,72: INK 0
170 PRINT PAPER 6;AT 19,0;" xmax"
180 PRINT PAPER 6;AT 21,0;" fix"
185 PRINT PAPER 5;AT 4,1;"*": LET a1=1: LET a4=4
190 GO SUB 950: GO SUB 960
200 PAUSE 50: LET b$="": LET pr=0: GO TO 1000
210 LET borra=0: DIM w(2): DIM m(15,3): DIM o$(15): DIM p(10): LET n$="": LET ind=1: LET jnd=1: LET knd=2: LET p(1)=1: LET num=0
330 FOR w=1 TO LEN b$: LET a$=b$(w)
335 IF a$="(" THEN LET p(knd)=ind: LET knd=knd+1: GO TO 500
340 IF a$="." OR (CODE a$>=48 AND CODE a$<=57) THEN LET n$=n$+a$: LET num=VAL n$: LET jnd=1: GO TO 500
345 LET n$=""
346 IF a$="-" AND w=1 THEN LET n$=a$: LET num=-1: GO TO 500
347 IF w<>1 THEN IF a$="-" AND NOT (CODE b$(w-1)>=48 AND CODE b$(w-1)<=57) AND b$(w-1)<>")" THEN LET n$=a$: LET num=-1: GO TO 500
350 IF a$="↑" OR a$="+" OR a$="-" OR a$="*" OR a$="/" THEN GO SUB 600: LET ind=ind+1: GO TO 500
360 IF a$="i" THEN LET num=num OR num=0: LET jnd=2: GO TO 500
370 IF a$="º" THEN LET m(ind,2)=num: LET jnd=2: GO SUB 4900: GO TO 500
380 IF a$="L" THEN LET m(ind,1)=num: GO TO 500
400 GO SUB 600: LET knd=knd-1: LET p(knd)=0: IF w=LEN b$ THEN GO TO 510
405 IF b$(w+1)=")" THEN LET num=m(ind,jnd): GO TO 500
410 LET a$=b$(w+1): LET o$(ind)=a$: LET m(ind,3)=1+(a$="*" OR a$="/")+2*(a$="↑"): LET ind=ind+1: LET w=w+1
500 NEXT w: LET a$="": GO SUB 600
510 IF VAL STR$ m(2,1)<>0 OR VAL STR$ m(2,2)<>0 THEN LET num=m(ind,jnd): GO SUB 600: GO TO 510
512 GO SUB 900: LET rp1=m(1,1): LET rp2=m(1,2): GO SUB 3300
580 PRINT AT 0,pr+1;"=";c$: BEEP .1,10: BEEP .1,5: LET w(1)=m(1,1): LET w(2)=m(1,2): GO SUB 800: GO TO 3000
590 FOR i=0 TO 31: PRINT AT 0,i;" ";AT 1,i;" ": NEXT i: GO TO 200
600 LET m(ind,jnd)=num: LET o$(ind)=a$: LET m(ind,3)=1+(a$="*" OR a$="/")+2*(a$="↑"): LET num=0
610 IF ind<>1 THEN IF m(ind,3)<=m(ind-1,3) AND p(knd-1)<>ind THEN GO SUB 100*CODE o$(ind-1)-5400*(o$(ind-1)="↑"): GO SUB 630
620 RETURN
630 LET m(ind-1,3)=m(ind,3): LET o$(ind-1)=o$(ind): LET ind=ind-1: GO SUB 610: RETURN
700 FOR k=1 TO d
705 LET z(k,3)=(z(k,2)-z(k,1))*(z(k,2)>z(k,1))+z(k,1): LET z(k,3)=(z(k,3)>71) OR (z(k,3)<=71): LET z(k,4)=(z(k,3)>1)
710 PLOT 183,72: DRAW INVERSE borra; INK z(k,4)+1,z(k,1)/z(k,3),z(k,2)/z(k,3): NEXT k: RETURN
750 FOR k=1 TO d: FOR j=1 TO 2: LET w(j)=z(k,j): NEXT j: GO TO 810
800 LET d=d+1: FOR k=d TO d: LET lastx=71
810 FOR j=1 TO 2: LET z(k,j)=w(j)*lastx/xmax: NEXT j: GO TO 705
900 FOR k=1 TO 2: LET m(1,k)=VAL STR$ (INT ((m(1,k)*10↑fix)+.5)/10↑fix): NEXT k: RETURN
950 INPUT "fix ";fix: BEEP .01,9: PRINT PAPER 6;AT 21,7;fix: RETURN
960 INPUT "xmax ";xmax: BEEP .02,9: PRINT PAPER 6;AT 19,7;TAB 12;AT 19,7;xmax: RETURN
1000 IF INKEY$="" THEN GO TO 1000
1010 LET a$=INKEY$: IF INKEY$=" TO " THEN LET a$="L": GO TO 1050
1020 IF INKEY$=" THEN " THEN LET a$="º": GO TO 1050
1040 IF a$<>"." AND a$<>"↑" AND a$<>"+" AND a$<>"-" AND a$<>"*" AND a$<>"/" AND a$<>"(" AND a$<>")" AND a$<>"i" AND NOT (CODE a$>=48 AND CODE a$<=57) THEN GO TO 1055
1050 LET b$=b$+a$: LET pr=pr+1: BEEP .02,9: PRINT AT 0,pr;a$
1055 IF CODE a$=12 AND pr<=1 THEN LET b$="": PRINT AT 0,pr;" ": LET a$="": LET pr=0
1060 IF CODE a$=12 AND pr>1 THEN LET pr=pr-1: LET a$="": PRINT AT 0,pr+1;" ": LET b$=b$( TO pr): PAUSE 50
1070 IF CODE a$=13 THEN BEEP .5,6: GO TO 1100
1080 IF a$=INKEY$ OR CODE INKEY$>=144 THEN GO TO 1080
1090 GO TO 1000
1100 GO TO 210
3000 GO TO 3000+(INKEY$<>"")*10
3010 LET a$=INKEY$
3020 IF CODE a$=13 THEN GO TO 590
3025 IF a$="f" THEN GO SUB 950
3030 IF a$="b" THEN LET borra=1: GO SUB 700: LET d=0: DIM z(10,4): BEEP .5,9: INK 6: PLOT 183,0: DRAW 0,144: PLOT 255,72: DRAW -144,0: INK 0
3040 IF a$="e" THEN LET lastx=xmax: GO SUB 960: LET borra=1: GO SUB 700: LET borra=0: GO SUB 750
3050 IF a$="m" THEN LET r(a1,1)=m(1,1): LET r(a1,2)=m(1,2): GO SUB 4950: GO SUB 900: LET r(a1,3)=m(1,1): LET r(a1,4)=m(1,2): GO SUB 4900: GO SUB 900: GO SUB 3200
3060 IF a$="c" THEN LET r(a1,5)=(r(a1,5)=0): GO SUB 3200
3070 IF a$="r" THEN LET w(1)=r(a1,1): LET w(2)=r(a1,2): LET borra=0: GO SUB 800: BEEP .2,15
3080 IF CODE a$=11 THEN PAPER 5: PRINT AT a4,1;" ": LET a1=a1-(a1<>1): LET a4=a4-3*(a4<>4): PRINT AT a4,1;"*"
3090 IF CODE a$=10 THEN PAPER 5: PRINT AT a4,1;" ": LET a1=a1+(a1<>5): LET a4=a4+3*(a4<>16): PRINT AT a4,1;"*"
3100 PAPER 7: GO TO 3000
3200 FOR y=1 TO 5: PRINT AT 3*y+1,3;"            ": LET rp1=r(y,1+2*r(y,5)): LET rp2=r(y,2+2*r(y,5)): IF r(y,5)=0 THEN GO SUB 3300: PRINT AT 3*y+1,2;c$: GO TO 3220
3210 LET c$=STR$ rp1: LET d$=STR$ rp2: PRINT AT 3*y+1,2;c$;"L";d$;"º"
3220 BEEP .1,20: NEXT y: RETURN
3300 LET c$=STR$ rp1: IF rp1=0 THEN LET c$=""
3310 IF rp1=0 AND rp2=0 THEN LET c$="0"
3320 IF rp2=0 THEN GO TO 3360
3325 IF rp2>0 AND rp1<>0 THEN LET c$=c$+"+"
3330 IF ABS rp2<>1 THEN LET d$=STR$ rp2: LET c$=c$+d$
3340 IF rp2=-1 THEN LET c$=c$+"-"
3350 LET c$=c$+"i"
3360 RETURN
3960 RETURN
4000 LET ind=ind-(ind<>1): GO SUB 4950: LET m(1,1)=LN m(1,1): LET m(1,2)=m(1,2)*PI/180: LET ind=ind+1: GO SUB 4200: LET m(1,1)=EXP m(1,1): LET m(1,2)=m(1,2)*180/PI: LET ind=ind-1: GO SUB 4900: LET ind=ind+1: LET m(ind,1)=0: LET m(ind,2)=0: RETURN : REM 
4200 GO SUB 4950: LET ind=ind-1: GO SUB 4950: REM producto
4210 LET m(ind,1)=m(ind,1)*m(ind+1,1): LET m(ind,2)=m(ind,2)+m(ind+1,2): LET m(ind+1,1)=0: LET m(ind+1,2)=0
4220 GO SUB 4900: LET ind=ind+1: RETURN
4300 FOR i=1 TO 2: LET m(ind-1,i)=m(ind-1,i)+m(ind,i): LET m(ind,i)=0: NEXT i: RETURN : REM suma
4500 FOR i=1 TO 2: LET m(ind-1,i)=m(ind-1,i)-m(ind,i): LET m(ind,i)=0: NEXT i: RETURN : REM resta
4700 GO SUB 4950: LET ind=ind-1: GO SUB 4950: REM division
4710 LET m(ind,1)=m(ind,1)/m(ind+1,1): LET m(ind,2)=m(ind,2)-m(ind+1,2): LET m(ind+1,1)=0: LET m(ind+1,2)=0
4720 GO SUB 4900: LET ind=ind+1: RETURN
4900 LET radio=m(ind,1): LET m(ind,1)=radio*COS (PI*m(ind,2)/180): LET m(ind,2)=radio*SIN (PI*m(ind,2)/180): LET num=m(ind,jnd)
4910 RETURN
4950 LET radio=SQR (m(ind,1)*m(ind,1)+m(ind,2)*m(ind,2))
4955 IF m(ind,1)=0 THEN LET m(ind,2)=SGN m(ind,2)*90: GO TO 4970
4960 LET m(ind,2)=ATN (m(ind,2)/m(ind,1))*180/PI-180*(m(ind,1)<0)+360*(m(ind,2)<0 AND m(ind,1)<0)
4970 LET m(ind,1)=radio: RETURN
4980 IF c$="0" OR c$="i" THEN LET ind=ind-1: RETURN
4990 LET m(ind,1)=num: RETURN
```

Expansión

Una nueva unidad de discos viene a engrosar el capítulo de periféricos en lo que se refiere al almacenaje y recuperación de datos, programas, etc. Este es precisamente el talón de aquiles del Spectrum que como se sabe, sólo tiene previsto la utilización del cassette o, en todo caso, del Microdrive.

UNIDAD DE DISCOS «TRITON QUICK DISK»

Presentada en España por PROEIN, S. A., esta nueva unidad tiene dos puntos a su favor: Su diseño compacto y su sorprendente precio, de tan sólo 39.500 ptas.

Como dato negativo podemos señalar la incompatibilidad con otros periféricos y la no prolongación del bus de expansión, salvo para la impresora ZX Printer.

Todo el sistema operativo del Triton Disk (T-DOS) se encuentra en el Firmware del Interface que viene incorporado, por lo que no es necesario cargar ningún software adicional.

Permite la conexión de hasta dos unidades de disco.



La unidad utiliza discos de 2,8 pulgadas

La sintaxis de los comandos es bastante similar a la utilizada por el Interface 1 y el Microdrive. Todos ellos pueden utilizarse en modo programas (excepto «COPY») con la salvedad de que necesitan estar en líneas de programa donde no haya ninguna otra información.

El más grave inconveniente que hemos encontrado es que el formateo se lleva a cabo en sectores de 2558 bytes. De esta forma, al salvar un programa o bloque de Código Máquina, la mínima longitud que ocupa es de 2558 bytes, y si se pasa, aunque sea en un solo byte, ocupará ya 5116 bytes (2 sectores).

El formateo del disco debe hacerse por las dos caras, ya que la unidad sólo tiene un cabezal de lectura.

El comando CAT permite visualizar una completa información sobre el contenido del disco:

Nombre del fichero, tipo, tamaño y dirección de inicio, así como la cantidad de memoria disponible.

Todas las operaciones del tipo SAVE van automáticamente acompañadas del correspondiente VERIFY, lo que repercute en el tiempo de acceso medio, que viene a ser de unos 10 segundos.



La carga del disco se efectúa por la parte superior.

MENSAJES DE ERROR DEL T-DOS

1. «Name too long»: El nombre del fichero excede de 8 caracteres.
2. «File not found»: No existe fichero con ese nombre.
3. «No Diskette»: No hay disco en la unidad, o la puerta está abierta.
4. «Wrong Station»: La dirección de posición no ha sido detectada.
5. «Time out Error» (Error de sincronización): Está leyendo de un disco no formateado.
6. «Over Run»: Más de 4 caracteres han sido recibidos sin la atención de la CPU.
7. «EOF Error» (Error en fin de fichero): Este fichero ha sido invadido.
8. «Disk Read Error»: Error de lectura.
9. «CRC Error»: Un fichero incorrecto está siendo leído de un disco.
10. «Write Protected»: Disco protegido contra escritura.
11. «Insuficiente Disk Space»: Insuficiente espacio en el disco.
12. «File Too Large»: Fichero demasiado grande.
13. «Verify Error»: Error en la verificación.

RESUMEN DE COMANDOS DEL T-DOS

1. FORMAT • <número de unidad>
   Función: Para formatear e inicializar un disco.
   Ejemplo: FORMAT • 1
2. CAT • <número de unidad>
   Función: Para examinar los ficheros existentes en un disco.
   Ejemplo: CAT • 1
3. SAVE • <número de unidad>;b; "<nombre de fichero>"
   Función: Para salvar un programa BASIC en un disco.
   Ejemplo: SAVE •1; b; "TEST 1"
4. SAVE • <número de unidad>; d; "<nombre de fichero>"; <nombre de variables>
   Función: Para salvar un fichero de datos en un disco.
   Ejemplo: SAVE •; d; "TEST 2"; n
5. SAVE • <número de unidad>;m; "<nombre de fichero>"; <dirección de comienzo>; <dirección final>
   Función: Para salvar un programa en código máquina en un disco.
   Ejemplo: SAVE • 1 m; "TEST 3"; 32500; 32520
6. LOAD • <número de unidad>; "<nombre de fichero>"
   Función: Para cargar un fichero desde un disco.
   Ejemplo: LOAD • 1; "TEST"
7. ERASE • <número de unidad>; "<nombre de fichero>"
   Función: Para borrar un programa de un disco.
   Ejemplo: ERASE • 1; "TEST"
8. COPY • <número de unidad> TO <número de unidad>; "<nombre de fichero>"
   Función: Para copiar un fichero desde un disco a otro.
   Ejemplo: COPY • 1 TO 2; "TEST 3"
9. COPY
   Función: Para formatear un disco en el drive 2, y después transfiere bloques de ficheros de la unidad 1 a la 2.
   Ejemplo: COPY

ESPECIFICACIONES TECNICAS

- Número de unidades por sistema: 2 (máximo)
- Capacidad de carga (formateado): 100 K (2 caras)
- Número de sectores (formateado): 20 (por cara)
- Número de ficheros por cara: 20 (máximo)
- Longitud del sector: 2,5 K bytes
- Formato: Bisync
- Densidad de registro: 4410 B.P.I.
- Método de codificación: MFM
- Velocidad de carga típica: 10 segundos
- Límites ambientales óptimos:
  a) Temperatura: 21 °C ± 5 °C
  b) Humedad relativa: 50% + 15% – 10%
- Consumo de potencia: 4,5 VA (típico)

COMPRESOR DE PANTALLAS

Miguel SEPULVEDA

Los aficionados que hacen algún que otro juego, y los profesionales que se dedican a ello, saben que en la memoria del ZX-Spectrum de 48 K de RAM, sólo puede almacenarse, como máximo, la información completa de 7 pantallas incluyendo la memoria donde se ubican los archivos de presentación visual y los atributos, quedando por tanto muy pocos bytes libres en memoria para meter el programa del juego.



F. L. Frontán

La utilísima rutina que hoy presentamos permite, en código máquina, comprimir los archivos de presentación visual y de atributos completos al número mínimo de bytes para después, almacenarlos en memoria; es decir, sirve para reducir en lo posible los 6912 bytes que ocupa la pantalla.

En un segundo artículo se tratará de la rutina en código máquina que expande el archivo comprimido que ya está en memoria, y lo repone, ya expandido, en los archivos de presentación visual y de atributos. También se tratará de los programas BASIC para manejarlos.

Antes de explicar con detalle cómo funciona la rutina, es necesario una pequeña introducción para comprender su filosofía.

Si consideramos una sucesión de bytes, cuyos valores sean por ejemplo:

Ø,Ø,1,1,1,1,2,3,5,5,5,1Ø,1Ø,1Ø,1Ø,
1Ø,1Ø,1Ø,1Ø,2Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,
Ø,Ø,1,1,1,1,1,1,1

podríamos utilizar algún algoritmo para almacenar estos bytes en memoria sin necesidad de ocupar una posición cada uno.

Si partimos de dos contadores, uno para el número de bytes consecutivos con distinto valor, y otro para los bytes con igual valor, el procedimiento es bastante simple.

En primer lugar veamos cómo son los contadores. El número máximo de bytes que podemos tener iguales o repetidos serán 6912 que es el número máximo de bytes de la pantalla. En hexadecimal este número sería 1BØØH. Para almacenar este valor necesitamos dos bytes, el byte más significativo primero y el menos significativo a continuación (al contrario de como se almacenaría un registro doble en memoria).

El bit más significativo del contador (bit 15) lo emplearemos como indicativo de que viene una sucesión de bytes desiguales (bit 15=Ø) o de que viene una sucesión de bytes iguales (bit 15=1). Cada contador consta de un registro doble, en el cual, dependiendo del valor del bit 15, nos indicará si el número que precede a este bit, se refiere a bytes iguales o desiguales.

Así en el ejemplo anterior, en memoria tendríamos almacenados los siguientes valores:

Ø,2,Ø,Ø,8Ø,4,1,Ø,5,2,3,5,5,5,8Ø,8,
1Ø,Ø,1,2Ø,8Ø,1Ø,Ø,8Ø,7,1

Antes de proseguir, hagamos una pequeña consideración; la cantidad de memoria necesaria para contabilizar los bytes repetidos, son tres octetos (dos para el contador y otro para especificar su valor) por tanto, no sería rentable considerar los bytes repetidos cuando son tres o menos; es mejor, en este caso considerarlos como bytes de valor no repetido.

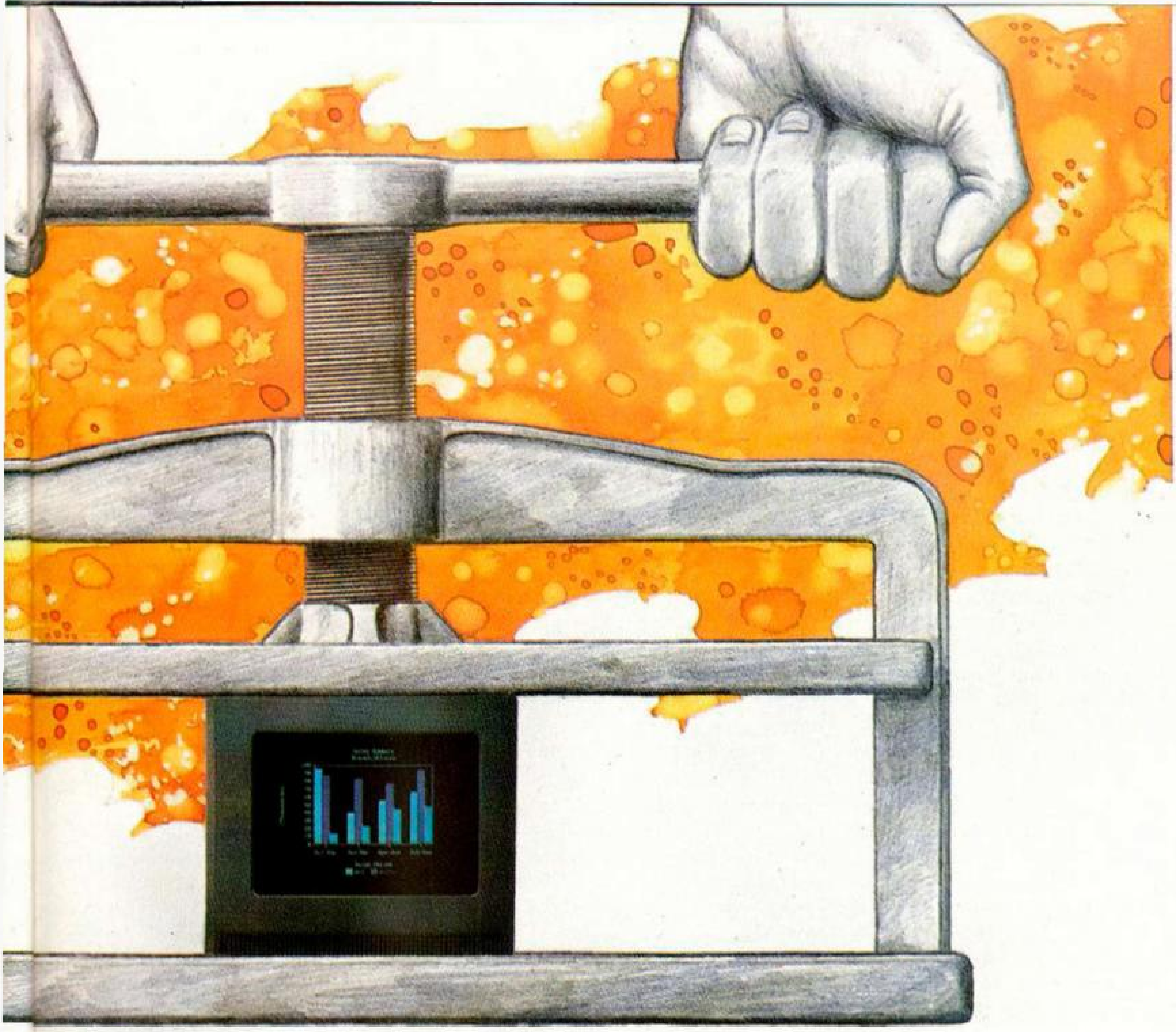
Veamos ahora el porqué de los números que hemos almacenado. Si consideramos la sucesión de bytes que pusimos como ejemplo, observamos que tiene en primer lugar dos bytes del mismo valor (que tendremos en cuenta por ser solamente dos, como si fueran de distinto valor); por eso, en memoria guardaremos Ø,2,Ø,Ø que nos indica: los dos primeros bytes son el contador (como el bit 15=Ø quiere decir que viene dos bytes de distinto valor) y los dos ceros siguientes son los valores de dichos bytes.

En la sucesión inicial a continuación hay cuatro bytes con valor 1; esto lo almacenamos de la siguiente forma: Contador de bytes iguales 8Ø,4 y valor de dichos bytes (1).

Luego vienen: 2,3,5,5,5 que consideramos de distinto valor y por eso almacenamos: Ø,5,2,3,5,5,5 siendo Ø,5 el contador de bytes distintos, y los demás números, los valores correspondientes a esos bytes.

Resumiendo, a cada sucesión de bytes distintos o iguales se les asigna unos valores en memoria. Así

VALORES REALES	*VARES…*
Ø,Ø ______	(Ø,[illegible],Ø…
1,1,1,1, ______	(8Ø…
2,3,5,5,5, ______	(Ø,[illegible]3,…
1Ø,1Ø,1Ø,1Ø,1Ø,1Ø,1Ø,1Ø ______	(8Ø1Ø…
2Ø ______	(Ø,Ø…
Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø ______	(8Ø),…
1,1,1,1,1,1,1 ______	(8Ø…



Los números entre paréntesis se refieren al contador.

Con este ejemplo, observamos que de una sucesión de 37 bytes ha quedado reducida en memoria a 26 bytes, lo cual supone un ahorro de aproximadamente, un 30% en este caso.

Funcionamiento del Programa

Las variables que usa esta rutina son para almacenar los contenidos de los registros HL, DE y el Acumulador.

VARIABLES QUE UTILIZA LA RUTINA	
23586– (REG1)	Indice de lectura de los archivos de presentación visual y de atributos.
23588– (REG2)	Indice de escritura en memoria de bytes comprimido.
23590– (REG3)	Valor de los bytes repetidos.

En la variable REG2 se guarda la dirección de memoria donde se almacena el archivo comprimido. En REG1 se guarda la dirección del primer byte del archivo (16384 D = 4000 H).

La rutina comienza inicializando a cero los registros DE y BC, que van a servir como contadores, el primero de bytes diferentes y el BC de bytes iguales. Esto se hace entre las líneas 90 y 190.

Entre las líneas 210 y 240 se comprueba si se han terminado de comprimir los archivos para continuar comprimiendo en caso de que no haya terminado, o para cerrar el archivo comprimido que se ha creado en memoria en caso contrario.

A partir de la línea 260 y hasta la línea 330, se leen bytes del archivo de pantalla de forma que si el byte leído es igual al siguiente, se incrementa el contador de bytes iguales y se guarda el valor del byte que se repite. Si el byte leído no fuese igual al que le precede, se pasa a ejecutar las instrucciones comprendidas entre las líneas 350 y 460 relativas al tratamiento de bytes desiguales: en primer lugar, se comprueba si se han leído más de tres bytes iguales, pues en caso contrario, no sería rentable hacer la compresión, y se tratarán como si fuesen bytes de valor diferente, incrementando por tanto el contador correspondiente a los bytes diferentes. Si el número de bytes leídos es mayor que tres, se bifurca a la parte de la rutina donde se hace la compresión y el almacenamiento en memoria. El

ALRES COMPRIMIDOS

0,20,0
30,1
0,52,3,5,5,5
30,10
0,10
30,0,0
30,1

LISTADO HEXADECIMAL

Linea	Datos	Control
1	210040ED5B765CED5324	991
2	5C22225C110000010000	270
3	7CFE5B301A7E23BE2006	932
4	0332265C18F078A7200B	777
5	79FE033006EB09EB1318	954
6	DEE52A245C7AB32810C5	1175
7	424BED5B225C70237123	890
8	EBEDB0EBC178B1280C78	1545
9	F680772371233A265C77	983
10	2322245CE17CFE5B38A7	1114
11	2A245C36FFED5B765CAF	1192
12	ED52444D03C900000000	668

proceso sigue hasta que no queden más bytes.

Entre las líneas 48Ø y 86Ø es donde se comprime los bytes y se almacenan en memoria. Lo primero que se comprueba, es si hay que almacenar en memoria bytes diferentes o bytes iguales. En caso de almacenamiento de bytes diferentes, se salva el contador de bytes iguales, se pasa al registro BC el contador de bytes diferentes, al registro HL la dirección de memoria donde se va a almacenar y al registro DE la dirección del archivo de pantalla desde donde se van a coger los bytes diferentes. A continuación, se almacena en memoria el contador de bytes diferentes, seguido de todos los valores de dichos bytes.

Tanto si ha habido bytes diferentes que almacenar como si no, se pasa a la parte de la rutina encargada de comprimir los bytes iguales y almacenarlos.

En esta última parte se comprueba si hay bytes iguales que comprimir y almacenar. Si los hay, se suma al cotador el valor 8ØØØH (es decir se pone el bit más significativo del contador a 1) para indicar que se trata del contador de bytes iguales. Esto es así, para indicar a la rutina que expandirá el archivo creado, que lo que a continuación viene es un valor repetido. Esto ya se verá posteriormente al tratar de la expansión de los archivos comprimidos.

Después de poner a 1 el bit más significativo del contador, se almacena en memoria el contador y a continuación, el valor del byte que está repetido.

Una vez almacenados tanto los bytes diferentes como iguales, la rutina pregunta si se ha terminado de comprimir todo el archivo de presentación visual y de atributos para cerrar el archivo en caso afirmativo, o para empezar otra vez desde el principio para seguir comprimiendo y almacenando.

Las instrucciones comprendidas entre las líneas 87Ø y 95Ø son las encargadas de cerrar el nuevo archivo de pantalla creado, para lo cual pone el valor FFH en el byte más significativo del contador de bytes diferentes o del contador de bytes iguales que en el primer caso, tendrá un valor máximo de 1BH y en el segundo caso, sería 9BH ya que al contador de bytes iguales se le suma 8ØØØH.

Los contadores de bytes se almacenan en memoria poniendo primero, el byte más significativo del contador y después, el menos significativo, al contrario de como se almacena un registro doble en memoria.

Termina la rutina hallando la cantidad de bytes que ocupa, restándole a la última dirección de memoria ocupada la dirección de carga del archivo comprimido en la varible del sistema SEED y devuelve el número de bytes ocupados en el registro doble BC, porque normalmente se llamará desde el BASIC y de esta forma imprimirá la longitud del nuevo archivo.

Esta rutina es reubicable, o sea, que se puede cargar en cualquier parte de la memoria RAM.

LOAD " " CODE dirección de memoria, 116

Para llamarla desde el BASIC se hace de la siguiente forma:

RANDOMIZE xxxxx: LET v = USR dirección donde se ha cargado. Donde xxxxx es la dirección de memoria a partir de donde se va a almacenar el archivo comprimido y v es el nombre de cualquier variable, la cual contendrá la longitud en bytes del archivo creado.

Este nuevo archivo se puede salvar en cinta de la siguiente forma:

SAVE "nombre" CODE xxxxx,v

y se puede cargar en memoria así:

LOAD " " CODE xxxxx ,v

La dirección de memoria donde se carga desde la cinta no tiene porqué ser la misma que donde se almacenó.

LISTADO ASSEMBLER DE LA RUTINA COMPRESORA

```
10
20          ORG  32768
30 ;
40 REG1     EQU  23586
50 REG2     EQU  23588
60 REG3     EQU  23590
70 ;
80 ;
90 PGSPM    EQU  $
100         LD   HL,#4000
110         LD   DE,(23670)
120         LD   (REG2),DE
130 ;
140 PPM0    EQU  $
150         LD   (REG1),HL
160         LD   DE,0
170 ;
180 PPM2    EQU  $
190         LD   BC,0
200 ;
210 PPM4    EQU  $
220         LD   A,H
230         CP   #5B
240         JR   NC,PPM10
250 ;
260 PPM6    EQU  $
270         LD   A,(HL)
280         INC  HL
290         CP   (HL)
300         JR   NZ,PPM8
310         INC  BC
320         LD   (REG3),A
330         JR   PPM4
340 ;
350 PPM8    EQU  $
360         LD   A,B
370         AND  A
380         JR   NZ,PPM10
390         LD   A,C
400         CP   3
410         JR   NC,PPM10
420         EX   DE,HL
430         ADD  HL,BC
440         EX   DE,HL
450         INC  DE
460         JR   PPM2
470 ;
480 PPM10   EQU  $
490         PUSH HL
500         LD   HL,(REG2)
510         LD   A,D
520         OR   E
530         JR   Z,PPM12
540         PUSH BC
550         LD   B,D
560         LD   C,E
570         LD   DE,(REG1)
580         LD   (HL),B
590         INC  HL
600         LD   (HL),C
610         INC  HL
620         EX   DE,HL
630         LDIR
640         EX   DE,HL
650         POP  BC
660 ;
670 PPM12   EQU  $
680         LD   A,B
690         OR   C
700         JR   Z,PPM14
710         LD   A,B
720         OR   #80
730         LD   (HL),A
740         INC  HL
750         LD   (HL),C
760         INC  HL
770         LD   A,(REG3)
780         LD   (HL),A
790         INC  HL
800 ;
810 PPM14   EQU  $
820         LD   (REG2),HL
830         POP  HL
840         LD   A,H
850         CP   #5B
860         JR   C,PPM0
870         LD   HL,(REG2)
880         LD   (HL),#FF
890         LD   DE,(23670)
900         XOR  A
910         SBC  HL,DE
920         LD   B,H
930         LD   C,L
940         INC  BC
950         RET
960         END
```

CONSULTORIO

Paginación de memoria

En el número 31 de MICROHOBBY respondéis a la pregunta de un lector diciendo que el microprocesador Z80 no puede direccionar más de 64K. Por el contrario, en la propaganda que aparece en algunos números de la revista sobre el ordenador Amstrad dice que éste utiliza dicho microprocesador, y que tiene 64K de RAM y 32K de ROM, que además son ampliables. ¿Cómo es esto posible?

Antonio L. ARETIO - Sevilla

☐ El microprocesador Z80 dispone de 16 líneas en el bus de direcciones, por lo tanto, su capacidad de direccionamiento inmediato es de 2 elevado a 16 = 65536 octetos, es decir, 64K.

No obstante, existe una técnica denominada «paginación de memoria», que consiste en dividir ésta en varios «bancos» a los que se accede de un modo selectivo. Esta técnica no sólo la utiliza el Amstrad, sino también el Spectrum 128K.

Avería del teclado

Tengo un Spectrum 48K comprado hace aproximadamente un año y medio y me han dejado de funcionar las teclas de los extremos (1, Q, A, Caps Shift, Ø, P, Enter y Space) bruscamente. ¿Podrían decirme a qué se debe? ¿Puedo repararlo yo mismo? ¿Presenta alguna ventaja el teclado profesional en este aspecto?

José A. CRIL - Cartagena

☐ Su problema podría deberse a una avería en la ULA, en cuyo caso, cambiar el teclado por uno profesional no resolvería el problema.

Otra posibilidad es que la avería esté en las membranas del teclado, en cuyo caso, lo más adecuado sería cambiarlo.

No obstante, lo más posible es que el error resida en la conexión del teclado con la placa de circuito impreso, concretamente, en la cinta de cinco conductores (la más extrecha de las dos). Le recomendamos que compruebe este extremo antes de tomar cualquier otra decisión.

Conexión vía radio

Somos un grupo de amigos aficionados al Spectrum y nos gustaría saber si sería posible conectar nuestros ordenadores para intercambio de datos y programas por medio de «radio-comandos». Este tipo de radio comando tiene un alcance de 20 km. aproximadamente (la mitad en ciudad) y nuestros Spectrums están separados unos 4 km., estos radio comandos tienen EAR y MIC y pueden conentarse al Spectrum. Nos gustaría saber si ésto es posible, ya que sería un fastidio comprarlos y que luego no sirviesen.

José A. CARRETERO - Oviedo

☐ La conexión de dos Spectrum vía radio es, cuanto menos, problemática; así lo decíamos a otro lector en nuestro número 42. Posteriormente, un lector radioaficionado nos escribió comunicándonos que esta conexión es perfectamente posible, ya que él lo venía haciendo hace tiempo.

En cualquier caso, los resultados dependen de un gran número de factores, por lo que no se puede afirmar con seguridad que el éxito esté garantizado.

Si deciden llevar a cabo la experiencia, les animamos a que nos comuniquen los resultados, así como los problemas que observen.

64 Columnas

¿Es posible mediante un programa en Código Máquina poder programar en BASIC con 64 caracteres por línea?

¿Se puede cambiar el modulador del video por otro para obtener mejor calidad de imagen, como por ejemplo la calidad que dan los ordenadores Canon, Hit Bit, etc.?

¿Cómo es posible que estos últimos, teniendo el microprocesador Z-80A puedan direccionar más de 64K de memoria?

R. MUÑOZ - Ciudad Real

☐ En el número 13 de nuestra revista publicamos un programa llamado EDITEXT que permitía imprimir a 64 columnas en Basic.

Puede sustituir el modulador de video por otro de similares características, pero es muy probable que no mejore la calidad de imagen.

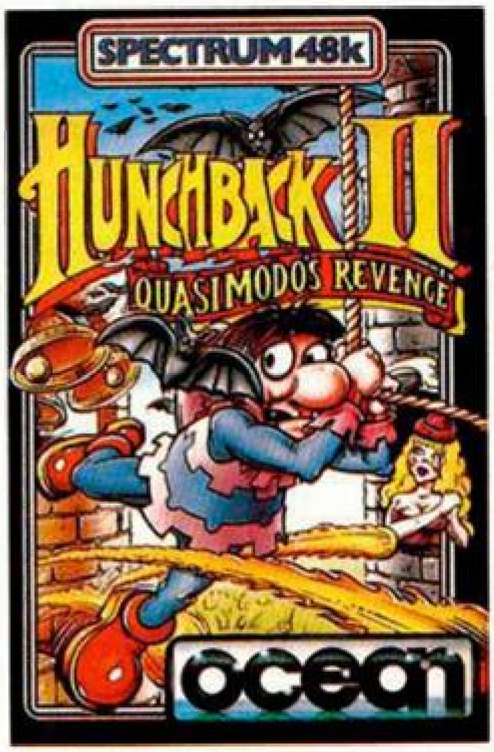
Los ordenadores que utilizan más de 64K de memoria total, con un Z-80, acceden a distintos bancos de forma selectiva; en otras palabras, tienen la memoria paginada.

Una oferta que vale por... 5

AL REALIZAR TU SUSCRIPCION A MICROHOBBY RECIBIRAS TOTALMENTE GRATIS UNA SELECCION DE FABULOSOS PROGRAMAS ELEGIDOS ENTRE LOS DE MAYOR EXITO DE TU FIRMA DE SOFTWARE FAVORITA: US GOLD

Blue Max
Tienes los controles de un avión de la primera guerra en tus manos. Derriba al enemigo, bombardea blancos, destruye tanques y emplazamientos de armas.



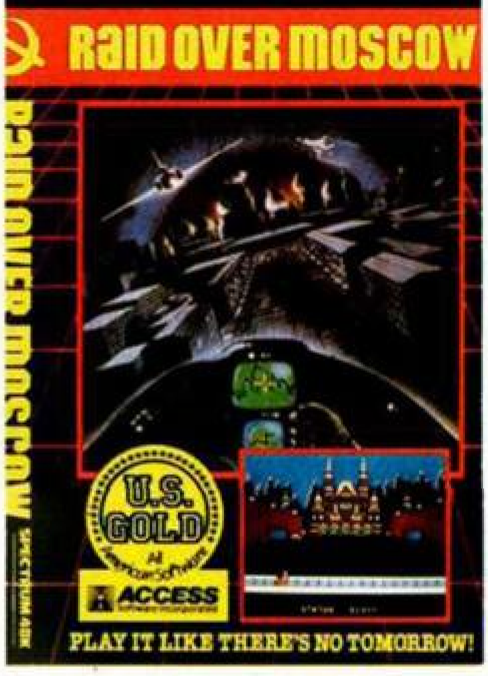


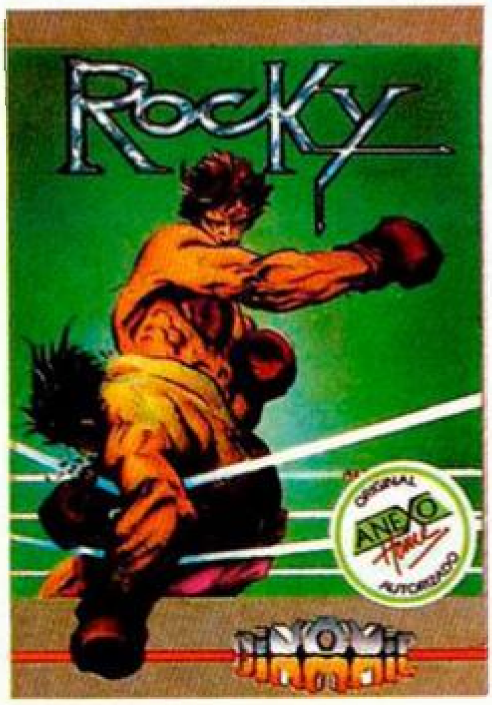
Hunchback II
Quasimodo busca a Esmeralda prisionera en el castillo. Durante las seis primeras imágenes debes accionar la campana gigante, hasta llegar a la liberación de Esmeralda.

Flak
En el año 2096 el universo está controlado por ordenadores. Una fuerza siniestra que quiere eliminar la libertad. Tu misión como piloto guerrero es atacar y destruir esta amenaza.



Raid Over Moscow
Defiende a USA y Canadá del ataque nuclear que ha lanzado Rusia contra ellos. Con tu escuadrilla habrás de hacer un viaje lleno de peligros hasta llegar al mismísimo Kremlim y destruir las bases de lanzamiento soviéticas. Gráficos y acción sensacionales.





Rocky
Vive la emoción de un encarnizado combate de boxeo, con Rocky. Tendrás que pelear duro para conseguir el campeonato del mundo; sólo los puños de tu oponente se interponen en tu camino hacia el triunfo final.

Estos cinco programas, que han encabezado listas de éxitos de toda Europa durante este año, están contenidos en **dos cintas** de cassette para Spectrum. De esta forma van a aparecer en breves días en el mercado inglés, como gran oferta de Navidad de la firma US GOLD, bajo el nombre genérico de **«Arcade Hall of Fame»**. En España se pondrán a la venta al precio de 2.500 ptas., aunque el valor total de los cinco programas es muy superior.

Microhobby Semanal te regala ahora estos cinco estupendos programas, al realizar tu suscripción, sólo hasta el próximo 31 de diciembre.

Lo increíble de esta oferta-regalo es que su valor es casi el 50 por 100 del precio real de suscripción.

Suscribirse a **Microhobby** es el sistema ideal para recibir nuestra Revista puntualmente y para ahorrar mucho dinero en su precio. Si consideras, además, el valor de los programas que recibes de regalo, no nos cabe duda de que preferirás recibir en tu casa **Microhobby** el próximo año.

ENVIA HOY MISMO EL CUPON QUE ESTA COSIDO AL FINAL DE LA REVISTA Y ELIGE LA FORMA DE PAGO QUE TE RESULTE MAS FAVORABLE

CONSULTORIO

Hardware y Código Máquina

¿Por qué el Spectrum no puede funcionar a 4 MHz? Si se pagina la RAM, ¿cuándo sabe el microprocesador a cual de los dos bloques de RAM debe acceder?

¿Cuál es la tensión entre los terminales de conexión Joystick que saca el Interface T - Kempston?

¿Cuánto tardaría aproximadamente un microdrive en cargar un programa de 48 K?

¿Qué pasa si se cambia el registro R del microprocesador?

¿Se puede hacer una ejecución paso a paso por soft., en Código Máquina, y cómo?

Javier ABAD - La Coruña

☐ El Specrum puede funcionar a 4 MHz, pero funciona a 3,5 MHz, para no llevar al Z-80 a su límite de funcionamiento.

Si se pagina la RAM, el microprocesador selecciona el «banco» al que accede mediante una instrucción OUT.

La tensión en los terminales del I. Kempston para joystick es de 5 V.

Un Microdrive tarda aproximadamente 5 segundos en cargar un programa; aunque depende más de lo que tarde en encontrarlo que de la longitud del propio programa.

Si se cambia el valor del registro R una vez, no tiene porque ocurrir nada. Si se le forzara a mantenerse continuamente en un determinado valor, una parte de la memoria quedaría sin regenerar.

En Código Máquina no se puede hacer ejecución paso a paso. Lo que sí se puede hacer es una «simulación de ejecución». Algunos monitores permiten intercalar «Breakpoints», que detienen la ejecución del programa y sacan los registros a pantalla.

"RANDOMIZEs" al azar

¿Tiene alguna utilización la sentencia RANDOMIZE USR 13Ø6?

¿Por qué al conectar el ordenador, salen durante unos segundos en la pantalla muchos colores?

Cesar GRANDE - Madrid

☐ La sentencia que nos indica, obliga al microprocesador a saltar en medio de la rutina de «salvar bytes»; en este caso concreto, se entra en medio de la rutina que salva un "1".

El saltar a la mitad de una rutina de la ROM sin haber fijado antes ciertos parámetros, tiene normalmente, consecuencias imprevisibles; aunque el resultado más posible es un «cuelgue» del ordenador (pérdida de control).

En el momento de conectar el ordenador, el contenido de la memoria es aleatorio, incluido el del fichero de atributos, es por eso que aparecen en la pantalla los colores que nos indica. Afortunadamente, en ese momento entra a funcionar la rutina de inicialización, que entre otras cosas, pone a "Ø" toda la memoria, con lo que se borra la pantalla.


WEST BANK
Defiende el
banco de Soft City
del ataque de los forajidos
y consigue escribir tu nombre
en la leyenda del «FAR WEST».
SPECTRUM 48 K y PLUS, 1.950 ptas.
Fotos tomadas de un Spectrum 48 K
OPERACION PUZZLE
Dinamic Software pone
en marcha la operación
puzzle: regalarte
6 millones de pesetas
en programas.
• Más información en los originales
DINAMIC
«MANSION DINAMIC»
c/ Tilos, 2, 21, MONTEPRINCIPE
Boadilla del Monte. MADRID
TEL.: 715 00 67
TIENDAS: 447 34 10

DE OCASION

● DESEO intercambiar copias de programas e instrucciones sin ningún fin económico. Enviad lista a la siguiente dirección: Jose M.ª Castañeda Vercher. Blasco Ibañez, 24. Señera (Valencia).

● CAMBIO amplificador Mistral 60 w, esfera del mundo con luz, calculadora Elite, 8002, tres revistas de fotografía y video, altavoz 2 w y Microscopio por Spectrum 48 K o Plus, o 35.000 ptas. Interesados escribir a Manuel Francisco. Eugenia de Montijo, 106, 2.º A. Madrid 28044. O llamar al Tel. (91)2081107.

● VENDO ordenador Plus con cables, y alimentador. Todo casi sin usar. Regalo una funda. El precio es de 35.000 ptas. Interesados llamar el Tel. (928)205829. Las Palmas. Francis.

● VENDO ZX Spectrum 48 K, interesados contactar con Antonio al Tel. (96)3518276 de Valencia (19 a 22 h).

● VENDO ZX Spectrum 48 k, incluido libros (Gráficos y Sonidos, Lenguaje Máquina, Basic, etc.) por 15.000 ptas. Interesados llamar a Javier. Tel.2421652 de Madrid.

● VENDO ampliación de memoria interna para ZX Spectrum modelo Issue 2, de 16 a 48 K, completamente nueva, sin usar, con garantía Indescomp, por 6.000 ptas. (vale 10.000) por no haber usado al cambiar de ordenador. Instalación muy fácil, no hay nada que sodar. Escribir a A. de la Llana. San Jacinto, 28. San Luis (Menorca).

● VENDO Spectrum 48 K, completamente nuevo y funcionamiento perfecto, con fuente de alimentación, cables, manuales, y cinta Horizontes (ambos en castellano) comprado en noviembre 1984, por el precio de 28.000 ptas. Interesados dirigirse a Fco. Javier Poyatos Aparicio. Avda. D. Juan Rodriguez, 5. Los Sarrios (Cádiz).

● AGRADECERIA a los lectores que me enviaran el código de acceso a los programas: Dragón Torc o Avalon, a cambio del código de acceso el Jet Set Willy, el mapa de Alien 8 o instrucciones de otros programas. Escribir a Manuel Martinez Moya. Plaza Carmen Benitez, 5, 3.º izquierda. Sevilla 41003.

● VENDO ZX Spectrum 48 K con todos los accesorios, 2 libros Basic en inglés y uno en castellano por 25.000 ptas. o bien lo cambio por Commodore Vic-20, con accesorios. Interesados llamar al Tel. (971)238048.

● VENDO ordenador «QL» nuevo en garantía con todos sus accesorios y además dos libros por el precio de 80.000 ptas. Tel. (93)3775534. Preguntar por Manolo (noches).

● VENDO instrucciones de la impresora Brother HR-5 o fotocopias de las mismas. Escribir a Manuel Freire Magariños. Avda. Sánchez Arjona. 39. 8.º B. Sevilla o llamar al Tel. (954)277528.

● VENDO ZX Spectrum Plus, casi nuevo, con transformador, manuales en español e inglés, cinta de demostración, dos cintas de Microhobby, 26 revistas de la misma, por el precio de 45.000 ptas. Llamar al Tel. 780356 o escribir a Jesús Jiménez Martinez. Bola, 40. Esquivel (Sevilla).

● VENDO consola juegos Atari, con cables, transformador, joystick compatible con el Atari y el Spectrum, otros dos mandos y 6 juegos, en perfecto estado y con los gastos de envío gratis. Escribir a Alexis Gutierrez. Gutierrez Rada, 2. Laredo. Cantabria. Precio: 25.000 ptas.

● VENDO Spectrum 48 K, casi nuevo, viene preparado con reset, piloto. Precio especial: 27.000 ptas. (negociables). Interesados llamar al Tel. (93)2043022. Ildefonso Lacosta Sansó. Manila, 51. Barcelona 08034.

● REGALO Interface joystick más de 100 revistas de informática (Microhobby, Micromania, Your Computer) por la compra de un ZX Spectrum 48 K, en perfecto estado por el precio de 25.000 ptas. También vendo Interface 1 sin usar por 8.500 ptas. Todo por 30.000 ptas., además regalo unos video juegos. O bien se cambia todo por otro ordenador (si es necesario, pago la diferencia). Llamar al Tel. (93)2186292 de Bacelona. Adriá Daniel Juliá Lundgren. Rambla de Prat, 9, 1.º 1.ª Barcelona 08012.

● VENDO ordenador Sinclair ZX-81d con sus correspondientes cables, manual de instrucciones y revistas Microhobby y Zx. El ordenador tiene fuente de alimentación y ampliación a 16 K. Todo por 16.000 ptas. Interesados llamar al Tel. (948)825828. Preguntar por José Antonio.

● VENDO joystick Quickshot II, totalmente nuevo, más Interface Kempston. Precio: 6.000 ptas. Llamar al Tel. (93)2249094. Preguntar por Eladio.

● VENDO Scalextric GP 51 por 9.500 ptas., y video juegos con 4 cartuchos. Las dos cosas por 18.500 ptas. (el precio del video es de 11.500 ptas.). Escribir a Antonio Luis Pérez González. Simón Ruiz, 3, 2.º A. Medina del Campo (Valladolid).

● COMPRO Ensamblador y compilador de calidad, con instrucciones. Enrique Serrano Expósito. Joaquín Benjumea, 52. Córdoba 14014.

● VENDO el siguiente lote: Joystick Gran Capitán más Interface tipo Kempston, 1 libro de programación Basic, 1 Your Computer, 3 Micromania, 5 Todospectrum, 19 Microhobby, 11 ZX. Todo por 7.000 ptas., contra reembolso. Adolfo Velasco Crespo. Avda. de España, 5. Ponferrada (León).

● VENDO ZX Spectrum 48 K, con cassette, joystick y libros de programación. Lo vendo suelto o con todo el lote. Urgente. Tel. (93)2336132. Antonio, (noches).

● CAMBIO ordenador ZX Spectrum 48 K, por ordenador MSX (Ram 48 K). También poseo abundante bibliografía. Interesados llamar al Tel. 4674814 de Madrid. Amador Merchán.

● VENDO ZX Spectrum 48 K, por el precio de 23.000 ptas. Pocas horas de uso, totalmente nuevo. Ponerse en contacto con David Fco. Franch. Rivero, 20, 2.º Barcelona. Tel. 3583926 (20 a 22 h).



SPECTRUM
PROGRAMAS AD HOC PARA TU DIVERSION

La mayor variedad de software de juegos pensados AD HOC para tu capacidad, tu habilidad y tu diversión.

Vídeo Pool

Bienvenido a Vídeo Pool, realista simulación del popular juego del billar americano.

Gran Casino

Juega al Bingo o la Ruleta sin necesidad de salir de tu casa. En familia o con amigos y mucho más divertido.

Brisca, Cinquillo

Los juegos de cartas de toda la vida, ahora en la pantalla de tu Spectrum.

Tute

Cántale las «cuarenta» a tu ordenador.

SPECTRUM.
Diversión Programada AD HOC

DISTRIBUIDOR EXCLUSIVO investronica

Tomás Bretón, 60. Telf. (91) 467 82 10. Télex 23399 IYCO E. 28045 Madrid
Camp, 80. Telf. (93) 211 26 58-211 27 54. 08022 Barcelona

etc

MAS ALLA DEL TIEMPO, MAS ALLA DEL ESPACIO...



DINAMIC

Dinamic Software presenta una nueva realizacion para Spectrum 48K · Plus · 128K. P.V.P.: 1.950 Ptas.

SGRIZAM. LA ESPADA DEL PODER

Tiendas y distribuidores Tel.: (91) 447 34 10. Pedidos contrareembolso. Tel.: (91) 715 00 67

¡¡¡Incluye operación puzzle: 6 millones de pesetas de regalo en programas. No te lo pierdas!!!